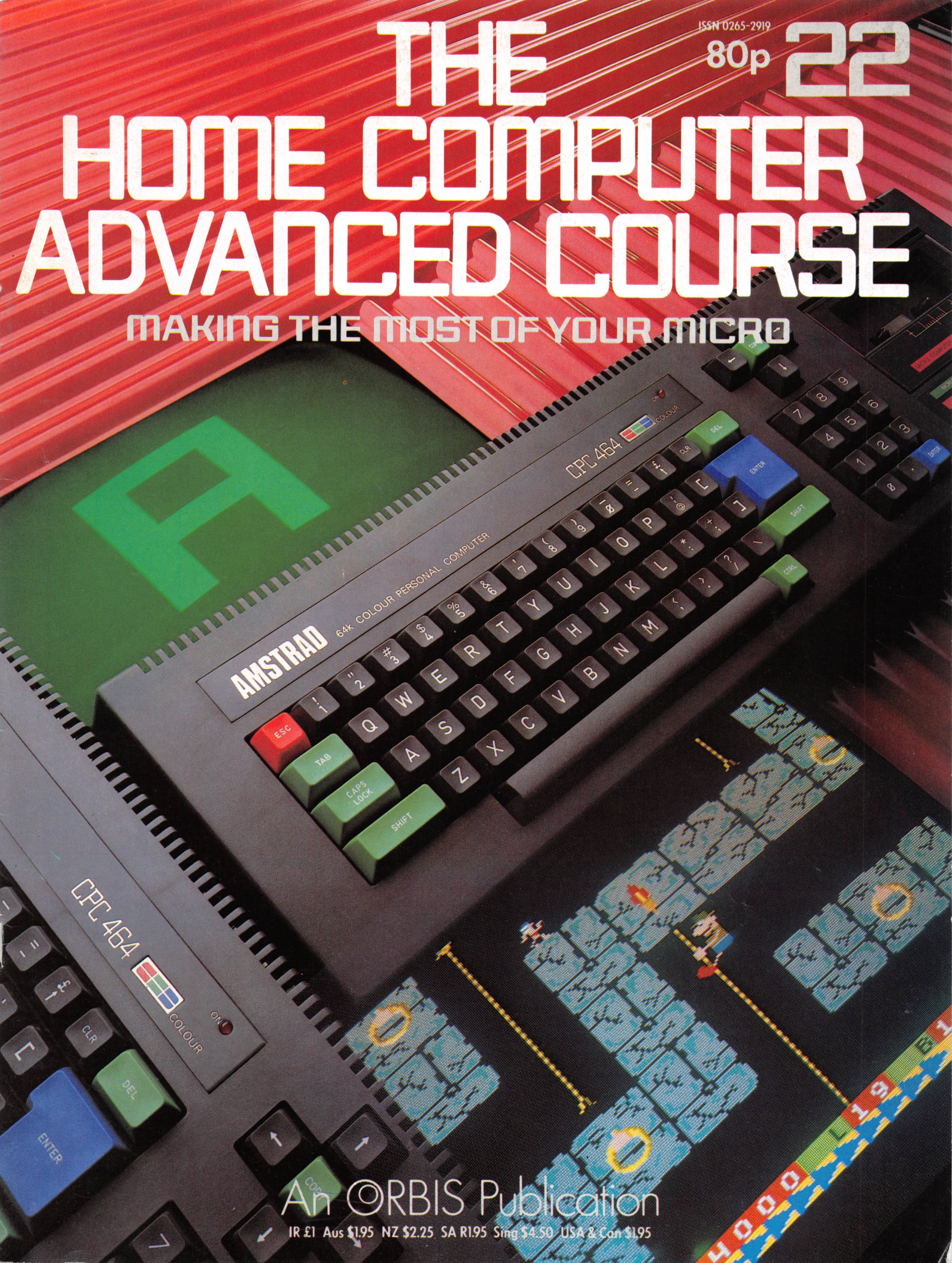
ISSN 0265-2919

80p 22

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO



An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- **We evaluate the Electron Plus 1 Interface from Acorn Computers, designed to bring the Electron closer to the BBC Model B. We compare an upgraded Electron to a BBC to find out**
- **The Psion Organiser is a new pocket computer from a company known for its software on Sinclair machines. We put the Organiser through its paces**
- **Bridge is a fascinating game that has recently been computerised. We look at programs designed for home micros and two dedicated machines**

QUIZ

1) Who is the MicroGnome?

2) In the BBC Minefield program what is the effect of pressing the cursor-down key on the variables delta-x and delta-y?

3) Which feature of the Amstrad computer allows the addition of an extra ROM chip?

4) Who invented the first integrated circuit?

Answers To Last Week's Quiz

A1) False, the EP-44 has a thermal ribbon allowing ordinary paper to be used.

A2) The strength of shot is determined by pressing the space bar.

A3) X co-ordinate 1,280 is beyond the resolution of the BBC Micro and has been included for future expansion.

A4) A decision table searches a series of conditions and acts when the condition is met. A decision tree acts on a condition and then branches to another condition.

QUIZ

COVER PHOTOGRAPHY BY CHRIS STEVENS

Editor Jim Lennox; **Art Director** David Whelan; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Art Editor** Claudia Zeff; **Chief Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Sub Editor** Steve Mann; **Researchers** Melanie Davis, Martha Ellen Zenfell; **Contributors** Steve Colwill, Steve Malone, Geoff Nairn, Geoff Bains, Matt Nicolson, Richard Pawson, Peter Jackson, Max Phillips, Steve Darroch; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Controller** Peter Taylor-Medhurst; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1P 1LB;

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

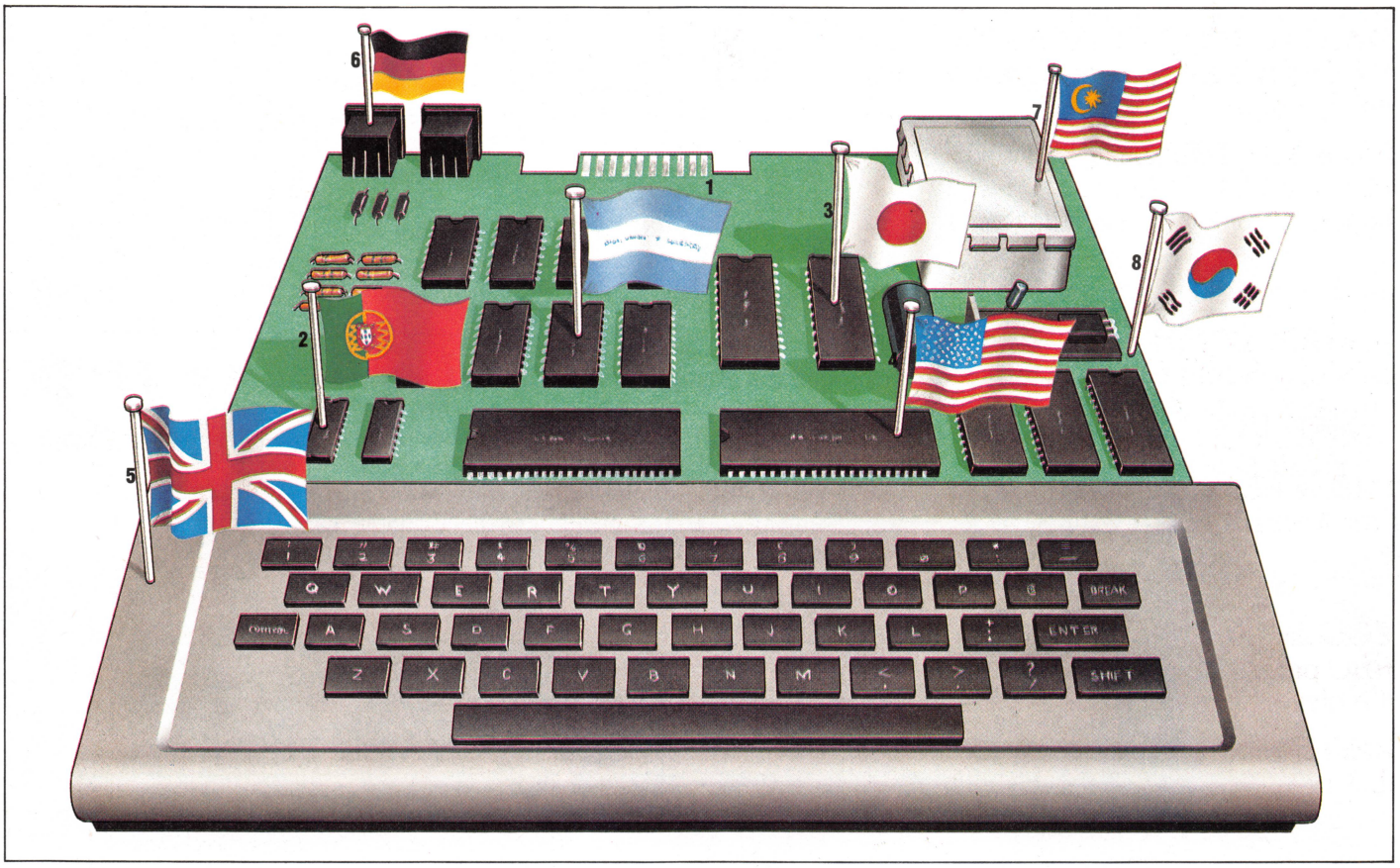
How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

WORLDLY GOODS



STEVE CROSS

The Melting Pot
This imaginary microcomputer is manufactured with parts from many different countries. Chips are produced in: 1) El Salvador; 2) Portugal; 3) Japan; 4) USA; case, keyboard, and final assembly are done in 5) UK; sockets are from 6) West Germany; the RF modulator is produced in 7) Malaysia, and the PCB board is manufactured in 8) Korea

Home computer manufacture is one of the few industries in which the company whose name is stamped on the case often has very little to do with the making of the product. Indeed, a close look at the origin of the components inside your micro may reveal an intriguing combination of different countries.

Making micros is a multinational business. The Amstrad CPC 464, featured on page 249, for example, is made entirely in Korea, and a large proportion of Acorn's BBC Micros are made in Hong Kong. Sinclair has always followed the policy of being strictly a research and design company, sub-contracting the manufacture of components and the assembly of the final machine to outside companies.

The reason for this is the need to make the machines as cheaply as possible. Manufacturing considerations are of prime concern to the computer's designers at the outset of the whole process. In order to keep costs down, the printed circuit board must be small and simple. This means that the design must incorporate as few chips as possible. This is not because of the cost of the chips themselves, but because fitting a large number of chips onto a board is expensive and can make the final product less reliable.

This last point is the reason for the use of ULA (uncommitted logic array) chips in most popular micros. The ULA, although an expensive chip to design and make, replaces dozens of smaller chip packages on the board.

The majority of microchips are made in California, where the term Silicon Valley has been coined to describe the area in which computer companies are concentrated. Once these chips are manufactured, they need to be encased in plastic or ceramic packages. This part of the process does not require the same degree of technical skill and is labour intensive. Since labour is cheaper outside the USA, the chips are shipped to various foreign countries for packaging.

When the board design is finalised, the search for a sub-contractor gets under way. Circuit board making, like chip making, is a complicated business, involving large investments in machinery, and there are many companies specialising in board manufacture. From detailed board blueprints, the board maker produces the

familiar green boards with metal stripes to link the electronic components.

Even the design of the board itself is subject to manufacturing cost constraints. It is possible to have multi-layer boards made, where several layers of metal are deposited with layers of insulation between them. However, this is expensive and is avoided by designers. Instead, a system of plated-through holes, in which holes for all wires are metal-plated inside to improve electrical contact, is always specified for computer boards because of the reliability this gives the finished product.

The power supplies, television modulators, connectors, keyboards and other components are bought in from all over the world, with cost being the main consideration. These parts then go to another sub-contractor, often overseas, for final assembly. Even the plastic case, which is made with expensive moulding machinery, comes in from yet another sub-contractor.

The final assembly of a computer can be done in two ways; either by highly automated means, or by a large number of cheap labourers doing the work by hand. The first option is generally the practice in the USA, Europe and Japan, while the second is common in Hong Kong, Singapore and Korea.

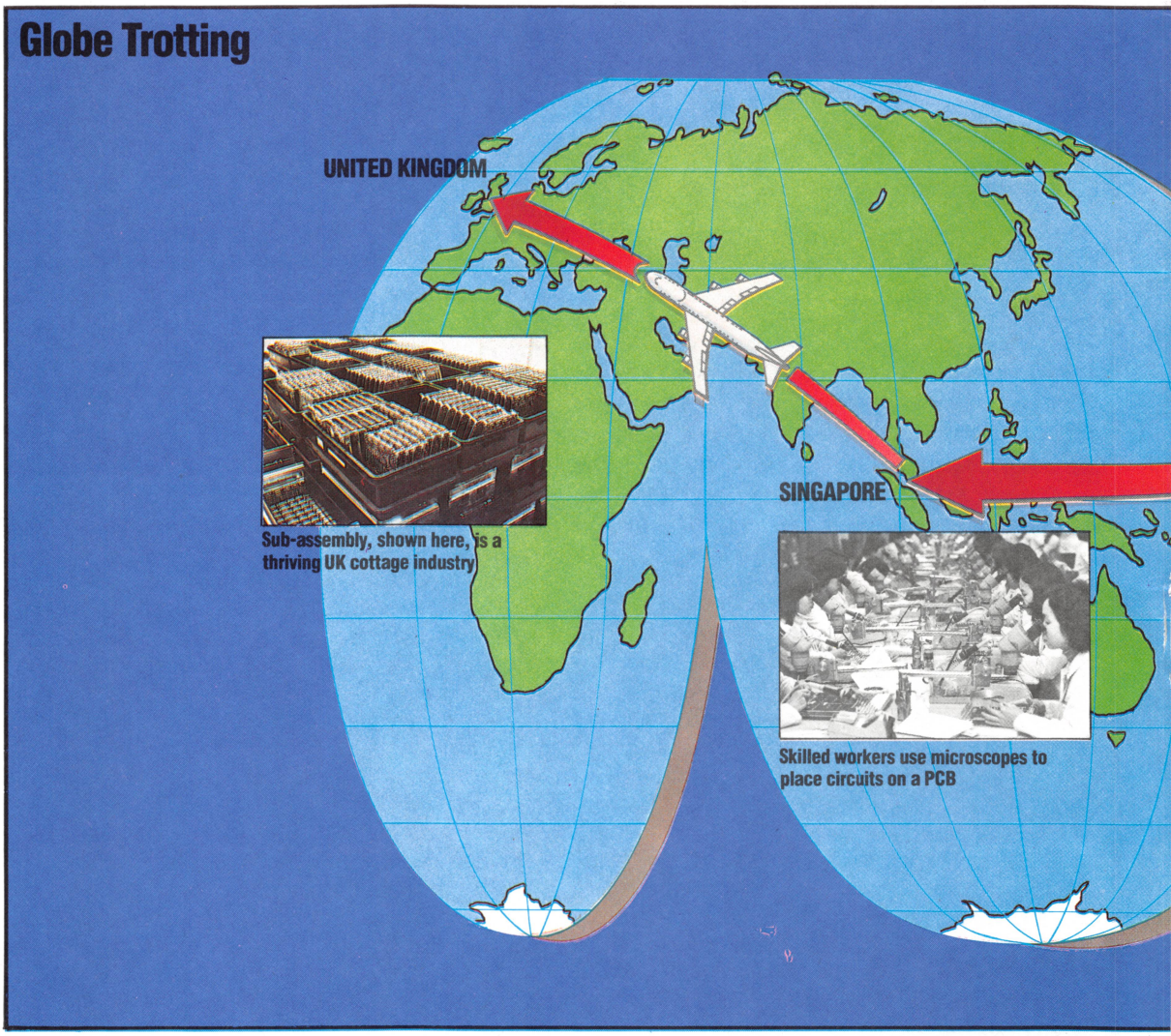
The automated assembly lines use robots to fit each component to the circuit boards. The robots are fed with bandoliers of components, ranging from capacitors to memory chips. All the operators need to do is to replace the bandoliers as they run out.

Whichever system is used, the boards are 'stuffed' with the appropriate chips and other components, with their leads protruding below the printed circuit board. The boards are then passed through a 'flow-soldering' machine that coats the protruding leads with solder. The solder is drawn up through the plated holes in the board and solidifies to give a reliable connection.

The completed boards are then tested, fitted into the cases, packed and shipped off to the warehouses for distribution, marketing and sale to the customers. This may sound simple, but every stage of the process has its own inherent problems.

SUPPLY LINES

The first problem is one of timing. All the components, from their various sources, must arrive in one place ready for assembly. The person responsible for this operation, the parts buyer, is one of the most vital links in the chain. He must have the ability to make a sharp business deal to



buy the parts cheaply, as well as scheduling and synchronising the delivery dates. All computer makers are aware of the disastrous effects of the late delivery of a single component. Idle production lines cost money and late deliveries will lose customers. Getting the best price for each component is also vital; a couple of pence on a connector can amount to a substantial sum of money in the high volumes that today's home micro makers are dealing with.

The assembly of the computer, whether by automation or by many human hands, is also an area that is prone to error. Components can easily be inserted upside down or back to front, or omitted altogether, completely ruining the final board. Flow-soldering can also miss one pin of a chip package. Similarly, some of the components in a batch may not meet their technical specification.

These problems explain the need for testing, both of the components and the finished boards. Many micro assemblers run spot tests on incoming components, and all of them run board tests of various levels of sophistication. Board testing is expensive, needing powerful computer hardware. But the investment has to be made: a sub-contractor won't keep his contract for long if machines fail to work on delivery.

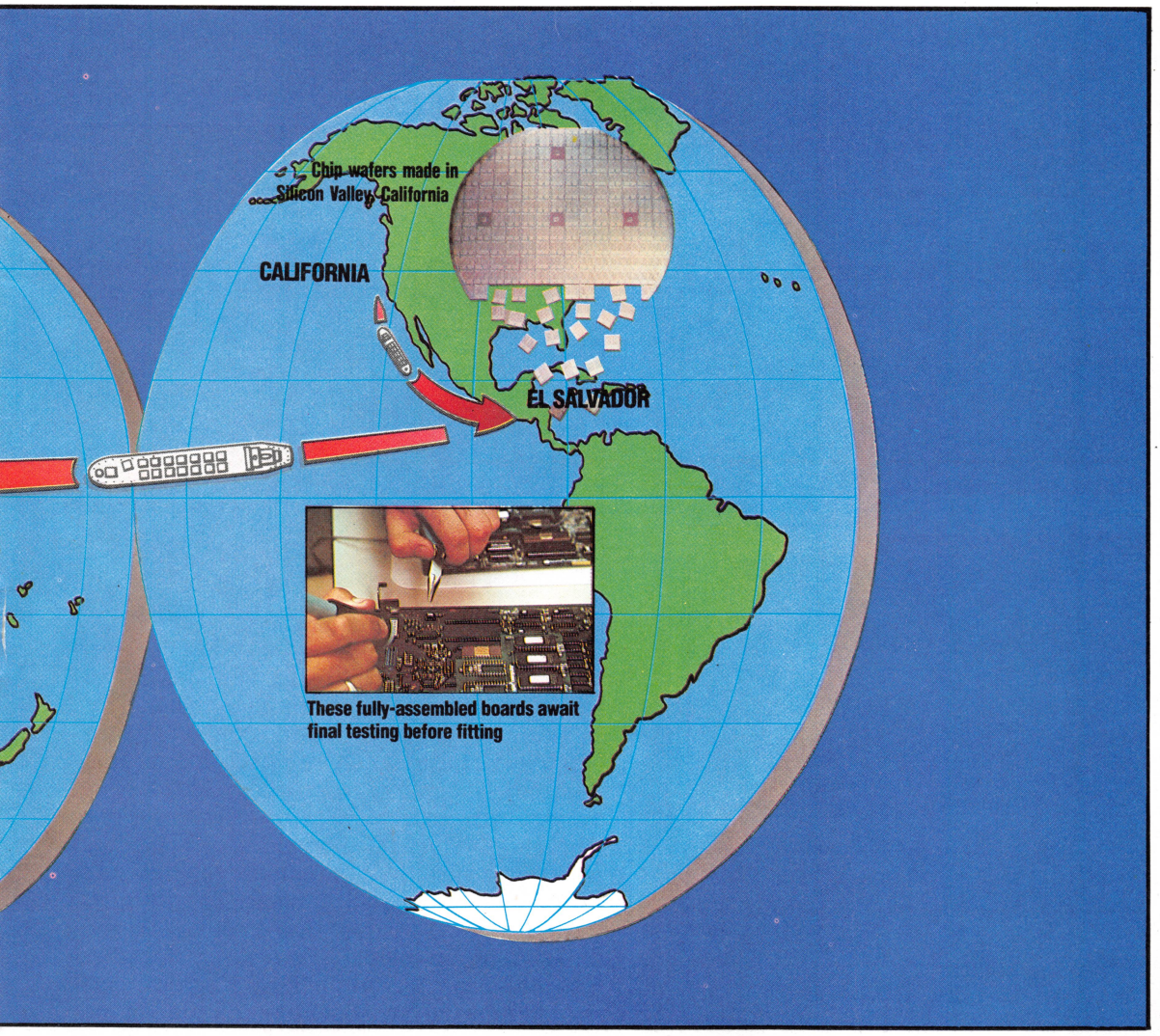
It is also common practice for the client to have a representative on the spot to test for faults in incoming parts and in the final product.

The sub-contractor that manufactures the Oric has devised an ingenious further test. The finished machines are individually weighed. If the machine is below the specified weight, then some components must be missing from the assembly. This is why each Oric box has a blue label quoting the machine's weight.

The final test of a computer is to plug the finished machine into a power supply and a television set. Makers of business machines often leave them running for a day or two 'soak testing', or 'burning in'. This simply involves leaving the machine running its built-in routines or accompanying software to ensure that everything is working properly.

With this number of variables, it is not hard to appreciate why home micros can be late or unreliable. The final assembler is dependent on chip and component suppliers to deliver on schedule and to the right specification. The design and marketing companies will depend on the final assembler to deliver the product in working order, and without keeping the customers waiting.

Cost Cutting
The purpose of transporting computer parts around the world is to save the manufacturer money and keep retail prices down. By using inexpensive labour resources in other countries, manufacturers have been able to reduce their costs significantly. But recent advances in automated production have made it possible to produce entire computers in Britain as inexpensively. The Oric Atmos, for example, is produced entirely in Britain, although Oric maintain production facilities in Singapore for their foreign markets



KEVIN JONES

Flying Components
This map of the world illustrates the movement of microcomputer components in the assembly process

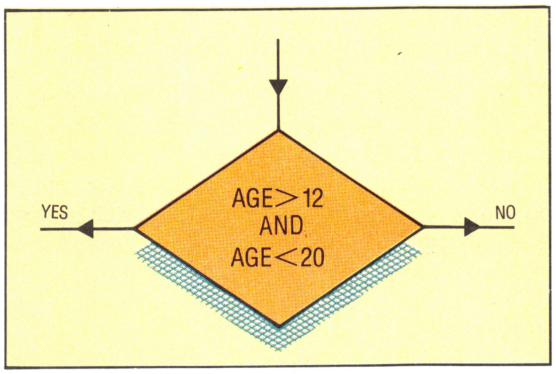
DECISIVE MOVES

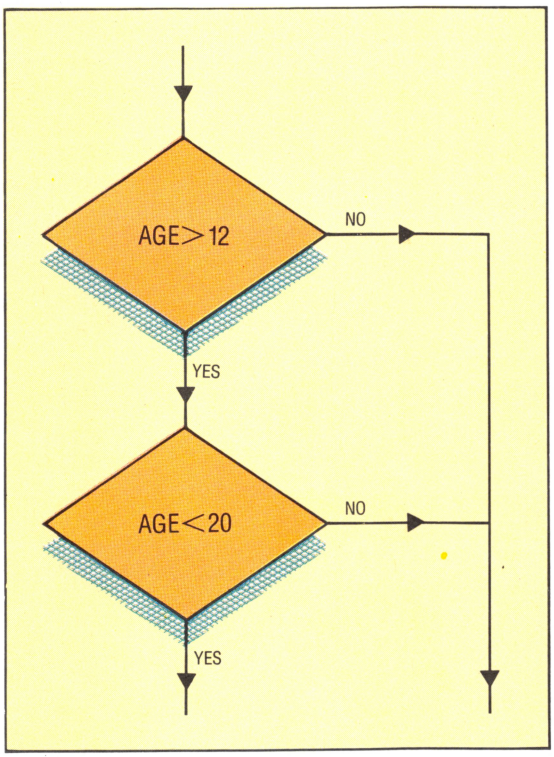
We continue our look at how to improve our use of flowcharts in the planning stages of program development. Here, we show you how compound decisions can be broken down into simple components, and look at the use of 'decision tables' in the more complex cases.

Programmers often want to use compound decisions in their programs such as:

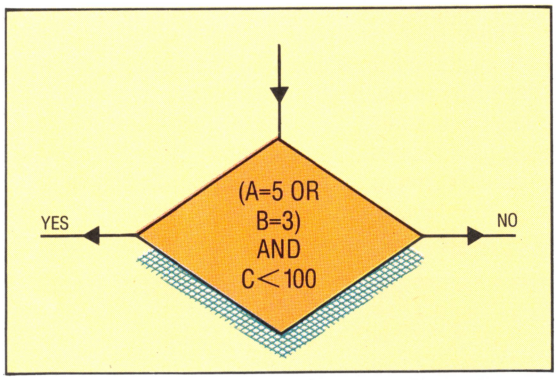
```
IF AGE>12 AND AGE<20 THEN STATUS=
  "TEENAGER"
```

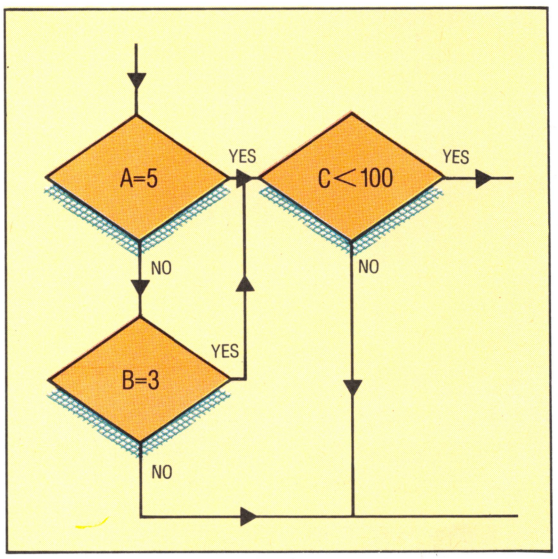
Algorithms with instructions like this are easier to understand if the compound decision is broken down into its component decisions.





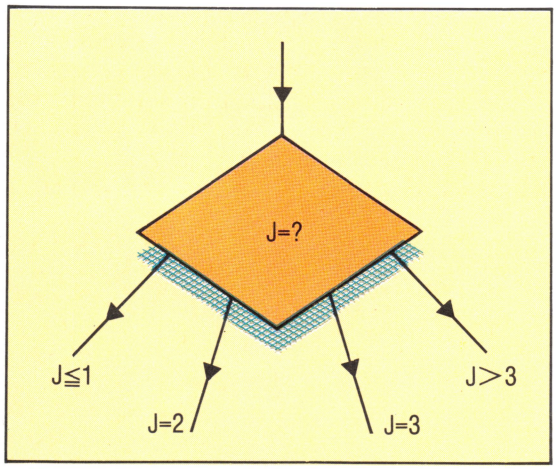
We have represented our BASIC example in diagrammatic form to show how the compound version is less satisfactory than the simple version. Our second example (below), consisting of three component decisions, makes the flow of logic through the decision boxes far more intelligible than its compound counterpart. It also makes clear a similarity with the rules of Boolean logic, which enable the construction of complex circuits out of a combination of simple logic gates.

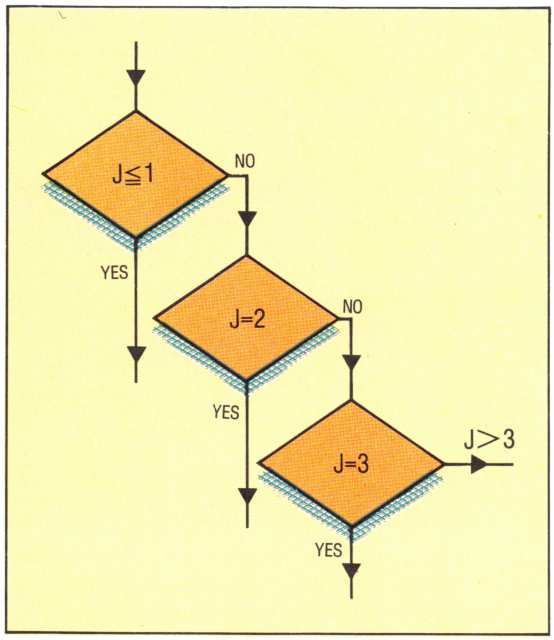




In the examples we have used, all the decisions have been binary ones, yet it is quite common for an algorithm to involve decisions with more than two possible outcomes. If we are reading an input from the keyboard that represents a selection from a menu, we would then want to branch to one of a number of different subroutines to take the requested action. To do this, most programming languages provide multiple branching constructs such as CASE..OF..in PASCAL and ON...GOTO and ON...GOSUB in BASIC. The rules for binary decisions also apply to multiple decisions: only one route may be taken out of the decision box and

all exits must be well labelled, with all possible routes being mutually exclusive and covering all possibilities. A multiple decision may be drawn, as in our example, with a set of exit paths leading from the same decision box. However, it is rare to see this and, more often, the decision will be broken down into binary decisions, as shown.





All multiple decisions can be represented as a set of binary decisions in this way.

THE DECISION TABLE

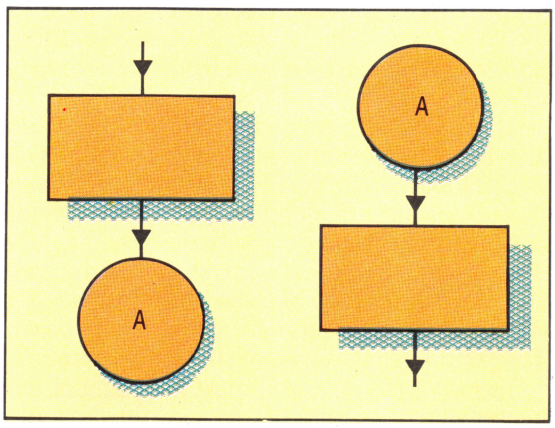
As an alternative to flowcharts, especially where there may be many multiple decisions, we can recommend the use of decision tables. We give an example of such a table, which represents a set of rules for making decisions. The table has four main sections: text describing the conditions for the rules, text describing the actions to be taken, a grid showing how the conditions fit into the rules, and a grid showing which actions are appropriate to each rule. In the 'conditions/rules' grid, values of variables appear in the cells, while in the 'actions/rules' grid below it, a tick indicates what action should be performed and a value acts as an input parameter for that action. Rule 4, for example, reads as: 'If the fire button is pressed and the game level is 2 and the player level is novice, then activate random blaster shields and reduce energy level by two per cent.'

	RULES							
CONDITIONS	1	2	3	4	5	6	7	8
FIRE BUTTON PRESSED	✓	✗	✗	✗	✓	✗	✓	✗
GAME LEVEL	1	1	2	2	1	1	2	2
PLAYER LEVEL	NOVICE	NOVICE	NOVICE	NOVICE	EXPERT	EXPERT	EXPERT	EXPERT
ACTIONS								
ALIEN EVASIVE ACTION							✓	✓
RANDOM BLASTER SHIELDS			✓	✓			✓	✓
REDUCE ENERGY LEVEL BY	1%	1%	2%	2%	2%	2%	4%	4%

Decision tables also serve to combine simple decisions into compound decisions and, in simpler forms than the one given here, are exactly equivalent to the truth tables used for predicting the output of logic gates.

One final point about the use of flowcharts. Wherever possible, restrict your flowcharts to one page. It can be irritating and time consuming leafing through many pages of paper. If your algorithm becomes too large, try and break it down into smaller algorithms. Remember that each algorithm can be used as a single instruction in some other algorithm. In this way, each routine in a program could be written as a single process box in a flowchart of the whole program, even if that routine uses other routines that in turn use others, and so on.

Inevitably something will go wrong now and then and a need will arise for a flowchart to continue beyond one page. If this happens, divide the flowchart at a suitable point (a decision, say) and use a circle with an identifying symbol inside it to point to the place where the flow of control continues on the next page (represented by another circle with the same symbol inside it, as shown below). If control returns to the main program, use the circles again to point back. Another solution is to view the missing portion as a separate algorithm, refer to it in a process box and represent it with its own separate flowchart.



LIZ DIXON

MAKING CONNECTIONS

The Prism VTX5000 modem is one of the most ingenious add-ons in the world of home computing. By linking the most successful home computer, the Sinclair Spectrum, to Prestel's Micronet and similar databases, it has opened the way for thousands of micro owners to step into the exciting new field of communications.

The Prestel database struggled to find users from the day it first went 'on-line'. Prestel 'sets' were expensive, and the amount of information available on the database was too limited to offer much of an advantage over an ordinary newspaper. However, with a suitable modem and software, most home micros can access Prestel. And so, in order to exploit this possibility, Prism formed Micronet, a separate area within Prestel dedicated to news and information about micros. Modems and software that enabled access to Micronet and Prestel were soon made available by Prism, and the project proved immensely successful. The number of users and the range of information services available continues to grow.

In producing an add-on to connect the Sinclair Spectrum to Micronet, Prism came up against a big challenge, as the machine is unsuitable for this application. It has no RS232 or serial interface, which means that ordinary modems cannot be connected. It has a screen display of 32 columns by 24 lines, and Prestel requires a 40 by 24 display, as well as the complicated 'teletext' graphics system. The company produced an all-in-one unit designed specifically for this single task: the Prism VTX5000 modem.

The unit contains all the interfacing needed to connect with the Spectrum, a direct-connect 1,200/75 baud modem and software to access Prestel. This not only provides the standard functions for logging into the system, but also uses the Spectrum's graphics screen to imitate a true 40 by 24 teletext display. As a result, for £100, Spectrum users can buy all the hardware necessary to join Micronet.

The VTX5000 sits underneath the Spectrum and connects to its expansion connector. The ribbon cable between the two has a third socket so that other Spectrum peripherals, such as a printer or Microdrives, can be connected. This unit plugs directly into the phone system rather than using an acoustic coupler (in which the telephone handset is pushed into two rubber cups on the modem). This provides much more reliable communication.

To install the unit, you must have one of the 'new-style' phone jacks. All recently fitted phones use these, but a suitable socket will have to be fitted in households with older telephones. You unplug the telephone, plug in the modem and then plug the telephone into the modem. This method avoids the need for a two-way phone jack on your wall, something that often costs an extra £10 with other modems. When connected, the telephone still works as normal.

When you switch on the Spectrum, it automatically runs the Micronet software. This is stored in ROM inside the modem so it doesn't need to be loaded before use. The Micronet package is very similar to those provided on other micros, so once you've used it, you will have no trouble using it on a different machine. The software is controlled by a series of menus and is simple to learn and use.

The first option on the menu is to 'log-on', which means keying in the 10-digit identity number given to each Prestel user. This number will be remembered by the computer as long as its power is on and so has only to be entered once

CHRIS STEVENS

each session, even if the user calls up Prestel several times.

The next step is to telephone one of the four computers that hold the Prestel database. When it answers with a high-pitched note, you press the Line switch on the modem and replace the receiver. The Spectrum is now 'talking' to Prestel. The first thing you must do is enter a four-letter password. This stops anyone else using your Prestel account and you can change it as often as you like to keep it a secret.

Once you have access to the database, it's like using any other Prestel adaptor. Prestel requires star (*) and hash (#) keys to move from page to page. The Spectrum uses Enter and Symbol Shift for the same functions. You can call up other functions of the Micronet software at any time, while still logged onto the system. These let you copy pages from the database, and store them on tape or print them out. Alternatively, you can 'download' (copy) whole programs off the system into your Spectrum. Micronet's telesoftware pages include free programs and programs that you pay for. The free ones are fine for a bit of fun, but you shouldn't expect too much for nothing.

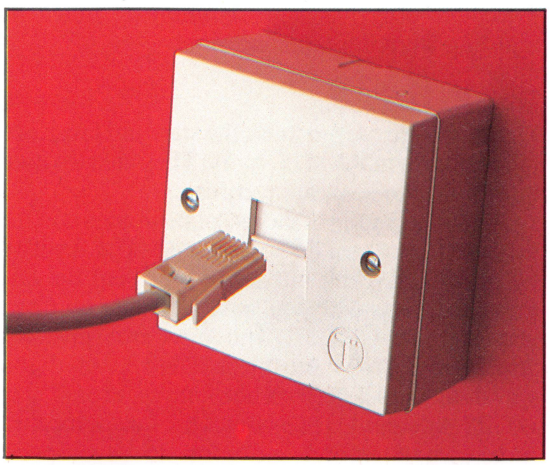
Both Micronet and Prestel offer a tremendous range of information. As well as news and reviews, there are pages of technical advice, jokes, games, letters, contact advertisements, and so on. You can even send electronic mail to other people who use the system regularly. Micronet also has a rival — the *Viewfax* newspaper in a different part of Prestel has an irreverent micro section run by the mysterious 'MicroGnome'.

The information 'explosion' on Prestel bodes well for the future. But some of its early problems are still apparent, particularly in Micronet. Many news pages are out of date and often the paths that lead from one page of information to the next are confusing. But if you tire of Prestel, you can try communicating with other Spectrum users directly. The VTX5000 is also designed to link two Spectrum computers via the telephone lines, so that they can send messages and programs directly to one another. The speed of transmission is a healthy 1,200 baud. Early versions of the modem were sold without the necessary software to do this, but now a tape that facilitates this data transfer is included with each unit. Obviously, this software is of use only to Spectrum owners who have friends who also have Spectrums and VTX5000 modems.

Several other communication standards are in common use for modems, but the VTX5000 cannot cope with them. The most important limitation is the modem's lack of the standard needed for the many free bulletin boards, which are being set up by computer communication enthusiasts around the world. These operate at a data rate of 300 baud, but the VTX5000 can't transmit as slowly as this.

Anyone who really wants to explore the whole area of computer communications with their Spectrum may find it more suitable to buy an RS232 interface (particularly Sinclair's own ZX Interface 1) and use a general purpose modem. However, this would probably entail writing your own software, wiring cables and so on. The VTX5000 is ideal for those who want to use Prestel only.

Direct Connect
The VTX5000 connects directly to the Spectrum with no additional interface required. Your telephone plugs into the VTX5000, which is then hooked directly into your modular telephone socket. Note that this modem will only work on modular jacks, not on older-style telephone connection boxes



BT Modular Phone Jacks
These jacks are being installed with all new phone lines, and conversions of existing lines. The jack takes a squarish plastic plug as shown here. If you do not have a modular jack, BT will install one on your existing line for £28 (excluding VAT)

D

DECREMENT

To *decrement* something is to reduce it in value, and in computing terms it usually means to decrease a figure by exactly 1. Thus, the variable A in BASIC is decremented by the expression:

```
LET A = A - 1
```

The most common use of decrementing is in counting loops. A value is decremented on each pass of the loop, and a condition is inserted to test for zero and terminate the loop. Most BASICs, however, feature several high-level structures for implementing loops more directly, such as FOR...NEXT, REPEAT...UNTIL and WHILE...WEND.

In machine code it is a different matter, and decrementing the value in a byte is so common that a special op-code is usually provided. DEC will take one from the value in a specified byte — a function that would otherwise entail half a dozen op-codes. DEX and DEY decrement the X and Y index registers respectively. The other main use of decrementing is in indexed addressing, where the program has to perform the same function on a whole sequence of stored bytes.

DEGAUSSING

After prolonged use, the read/write head on a tape recorder gradually becomes magnetised. Eventually, this magnetic build-up becomes serious enough to cause permanent distortion of the signal stored on the tape. Hi-fi enthusiasts have traditionally coped with this problem by using a *degaussing unit*, which is merely another name for a de-magnetiser.

For the home computer owner who relies on a tape recorder for program storage, tape care is extremely important. Tape heads and drive mechanisms should be cleaned regularly to remove the deposit of iron oxide that accumulates with regular use, and if a degaussing unit is available it should be applied to the tape head at intervals of a few months. These units are usually the size of a small torch with a metal probe at one end. They are mains-powered and cost only a few pounds. In use, the metal probe is placed on the read/write head, the power switched on, and the probe slowly withdrawn to a distance of a few feet before the power is switched off. This is an extremely effective way of removing magnetic build-up from the tape machine.

DELIMITERS

Delimiters are used to mark the start and finish of a block of data in RAM memory. They are also used in disk files to separate records.

A good example of the use of delimiters is in the way BASIC program lines are stored. The first two bytes form a 16-bit integer to represent the line number (this is why you can't have line numbers greater than 65,535). The next two give the address in RAM where the next line is stored; this speeds up the search for a line number in a GOTO or GOSUB statement. Then comes the text of the line, usually in a compressed form to save space. Finally, there is a delimiter, which is frequently a byte containing zero, to signify the end of the line.

It is important that the delimiter value does not appear as part of the data. In the above example, zero cannot appear in the text of the line. This is not a problem, because if a 0 features in a line like:

```
B = A*10 + 2
```

the 0 will be stored as its ASCII value, which is 48.

DIAGNOSTIC ROUTINE

The idea of a computer being able to tell you if it is malfunctioning is a very appealing one, though it would destroy the fundamental plot of many science fiction novels! That's exactly what a diagnostic routine does. Sadly, few computers feature diagnostic routines, but those that do range from micros to mainframes.

No system can be totally self-diagnosing. If the power supply has failed, this won't be diagnosed, because the diagnostic routine can't run. Diagnostic routines can, however, check the operation of all of the memory, interface chips and main logic boards in the system. Most micros feature a very simple routine that checks out the main RAM by storing a value in each byte and then reading it back again. This serves not only to identify any faulty locations, but to ascertain how much RAM has been fitted to the system.

The best example of diagnostics on a microcomputer is on the DEC Rainbow. Each board is fully tested by software, and, if a fault is identified, the location of the unit or board is shown graphically on the screen. The physical design of the Rainbow is such that any board can be removed and replaced with ease.

IAN McKINNELL

Diagnostic Routine
The Newbrain microcomputer fills its one-line display with 'hieroglyphic' characters while running its self-test diagnostic routine

DIGITAL

A *digital signal* is one whose voltage at any particular time must be one of a group of discrete levels. In computing, there are two such levels — each bit in the computer's memory may be considered as a switch that is either on or off. In contrast, an analogue signal is one in which the range of possible values varies continuously.

PACKAGE DEAL

The Amstrad CPC 464 packs more computing power and facilities into a cheaper package than most home computers. Not only do you get a sophisticated home micro, but a built-in cassette recorder and a display monitor are supplied as standard.

The Amstrad computer is available in two versions. Both models feature the same basic computer, but are supplied with different monitors — one monochrome, the other RGB colour. The power supply for the computer is housed inside the monitor case and connecting leads carry the power and monitor signals to the computer. As the cassette recorder is also built into the computer case, there is only a single mains power lead to supply the whole system, resulting in a minimum of connecting leads.

The monochrome monitor has a green screen that gives a very clear, crisp display, which is suitable for business and other textual work. There is a slight undulation of the picture, which is probably caused by the proximity of the power supply in the monitor case. The colour monitor is of medium resolution only. This means that, although this monitor will display the Amstrad's multi-colour graphics to the full, it cannot display 80-column text in a readable form.

To go with the businesslike monochrome monitor, the Amstrad has a full typewriter-style keyboard with a separate numeric keypad. All the keys may be redefined to create characters other than those marked on the key-tops. In addition, the numeric keypad can act as a set of programmable function keys. Any string of up to 32 characters can be programmed into each of these keys. Single keys can thus be used to load or list a program or to clear the screen.

The cassette recorder is fairly standard in design. The computer has simple stop-start control over the drive motor when a program is being loaded or saved. Two program loading speeds — 1,000 or 2,000 baud — are available on the Amstrad; these can be selected from the software. When loading a program, the computer automatically detects which speed was used for the recording and adjusts accordingly. Although the reliable 1,000 baud speed is used as the default setting, the fast speed is perfectly secure for most uses. This is the speed used by most of the program tapes available for the Amstrad.

As well as the features built into the Amstrad, there is also provision for a wide range of peripherals. A Centronics printer will plug straight into the micro via a rather crude edge-connector. Disk drives may also be added, but these are not yet available, although a disk drive add-on giving the computer extra memory, the

CHRIS STEVENS

Monitor Options
The cheaper version of the Amstrad comes with a monochrome monitor, but a version with a colour monitor costs under £350. The machines cannot be bought without monitors. The two programs seen running are Wordhang, which is a version of Hangman, and Admiral Graf Spee, a World War Two naval game

LOGO language and the CP/M business operating system is promised shortly. The Amstrad has a single joystick socket that accepts Atari-style joysticks. Some games need two joysticks, so Amstrad supplies a pair of joysticks that fit into the socket at the same time.

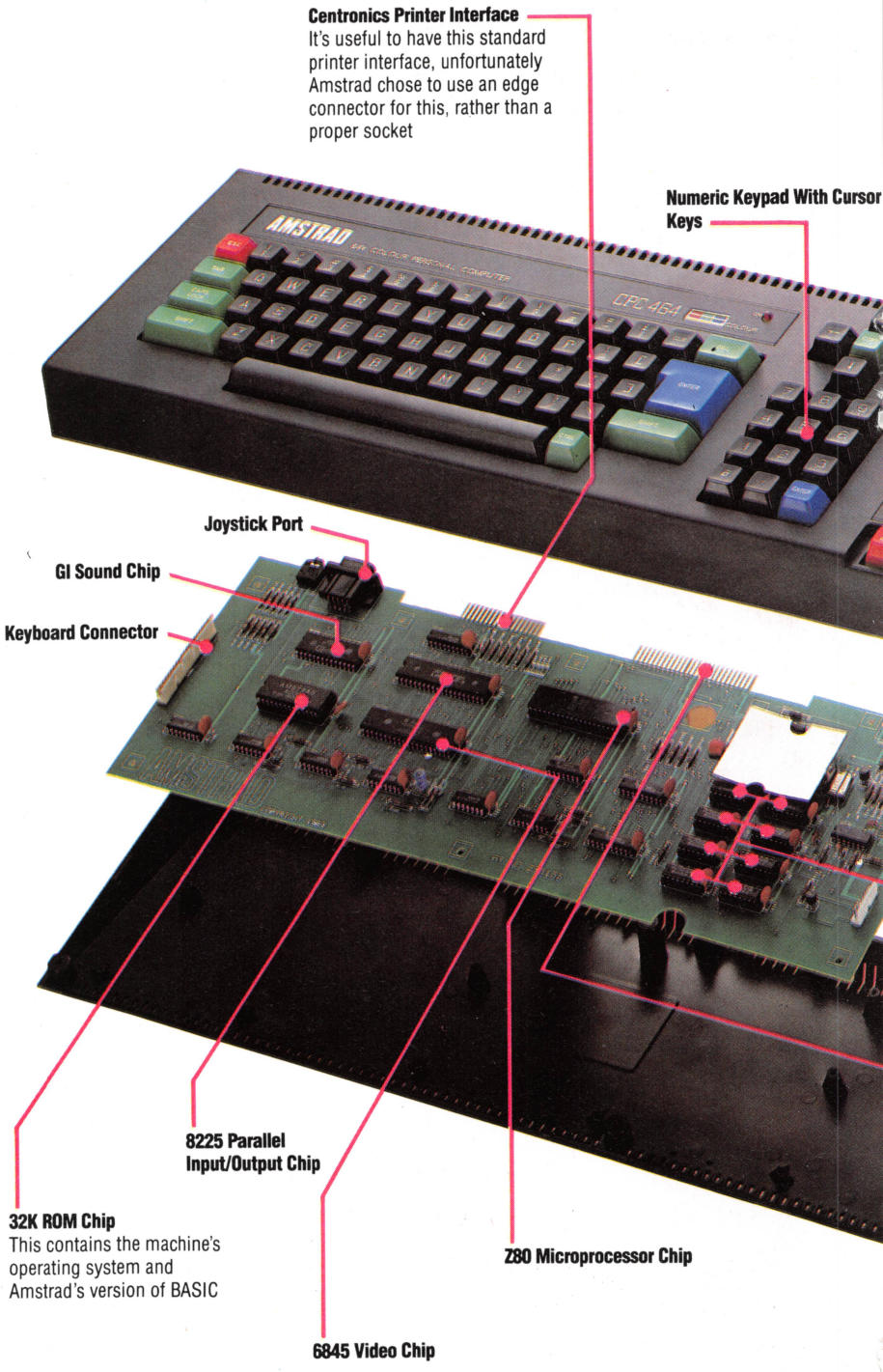
The Amstrad has a built-in loudspeaker (complete with volume control) but the sound may also be fed to an external amplifier to give stereo sound from the computer's three separate 'voices'. One voice is fed to the right-hand speaker, one to the left, and the third is mixed equally between them. With suitable programming, an alien spaceship can not only be made to move across the screen but its sound will appear to swoop around the room.

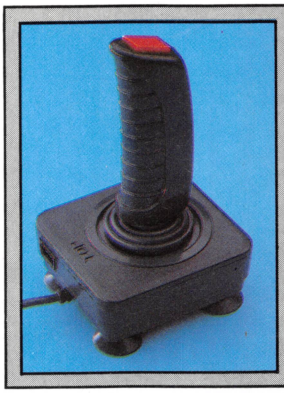
Controlling the sound from BASIC is very simple, despite the sophistication of the system. Notes can not only be made to start and stop at a chosen pitch and volume, but their envelope, or 'shape', can also be controlled. The volume envelope may be adjusted to make a note imitate the sound of, say, a piano or bell. The pitch envelope can also be controlled independently for creation of sound effects like sirens and whistles, while each sound voice can be mixed with 'noise', making explosions and gunshots easy to simulate.

The Amstrad's best feature is its superb graphics. There are three display modes available, each providing a different number of characters and a different number of colours on the screen at once. Each of these modes uses the same 16 Kbytes of memory and there is a trade-off between the number of colours and the resolution and text format. The highest resolution mode allows only two colours on the screen at any one time, one foreground and one background. 80-column text is available, and the machine supports an impressive 640 × 200 graphics resolution. At the other extreme, the 20-column mode gives 16 colours on the screen at once. The third mode allows four colours and 40 columns.

Although the number of colours on the screen at one time is limited, these can be selected from a palette of 27. Browns and pastel shades may be displayed as well as the usual red, green, blue, etc. The border colour around the display area can be selected from the same palette of colours. Any of the colours can be made to flash between two different shades at varying rates. As well as providing an excellent easel for detailed still pictures, the Amstrad graphics also provide a good base for animation techniques. Sprites are the only omission. However, the screen handling from BASIC is fast enough to make this deficiency less serious.

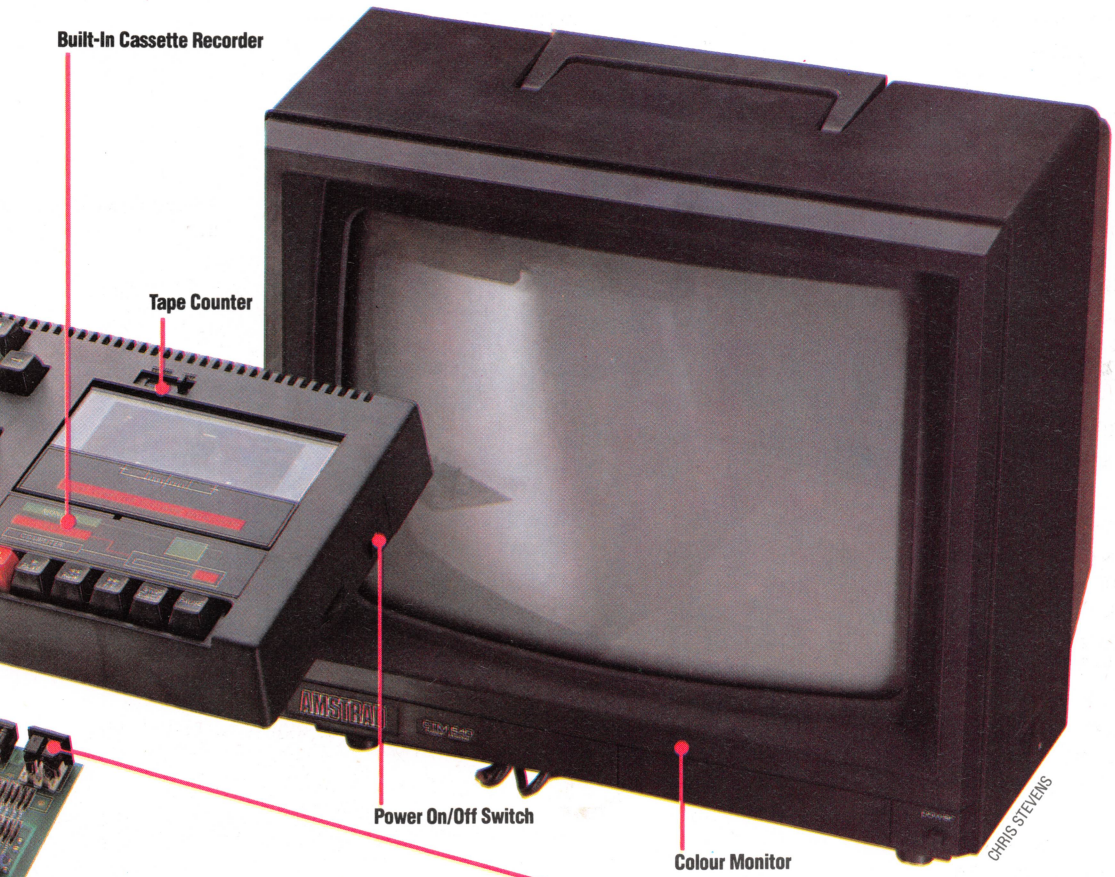
Although the screen display uses up a hefty 16 Kbytes of memory, this does not take memory space away from the user's program. The screen RAM and the BASIC ROM occupy the same area in the processor's memory map. A custom chip inside the Amstrad switches between the two as required, so a full 42 Kbytes of memory are left

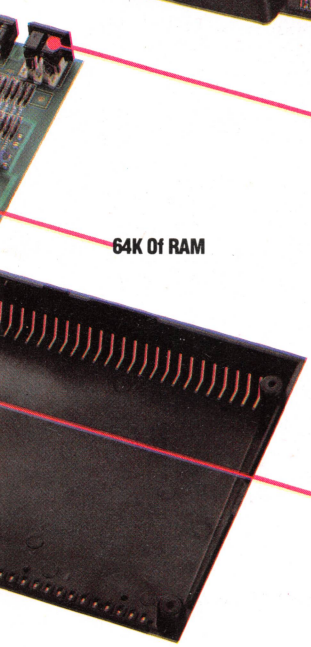




Juggling Joysticks

The Amstrad has one joystick socket that allows a standard Atari-type joystick to be used. However, the company also produces its own joystick, and this has a joystick socket on it that enables a second joystick to be plugged in. The signals from the second joystick are relayed through the first





Monitor Output
This socket gives out two signals: one for a monochrome monitor and one for an RGB colour monitor. Whichever type of monitor is bought with the Amstrad will be plugged in here

AMSTRAD CPC 464

PRICE
£239 with monochrome monitor
£349 with colour monitor

DIMENSIONS
Keyboard/cassette:
565x170x70mm
Colour monitor: 380x350x350mm

CPU
Z80

MEMORY
64K RAM, of which 42K is available for BASIC programs, 32K ROM

SCREEN
Three modes with full mixing of text and graphics:
640x200 (2 colours)
320x200 (4 colours)
160x200 (16 colours)
A palette of 27 colours to choose from

INTERFACES
Joysticks (2), Centronic printer port, expansion bus (disk drives), stereo sound output, monitor output

LANGUAGES AVAILABLE
BASIC (included), PASCAL (on cassette)

KEYBOARD
Typewriter-style, 74 keys

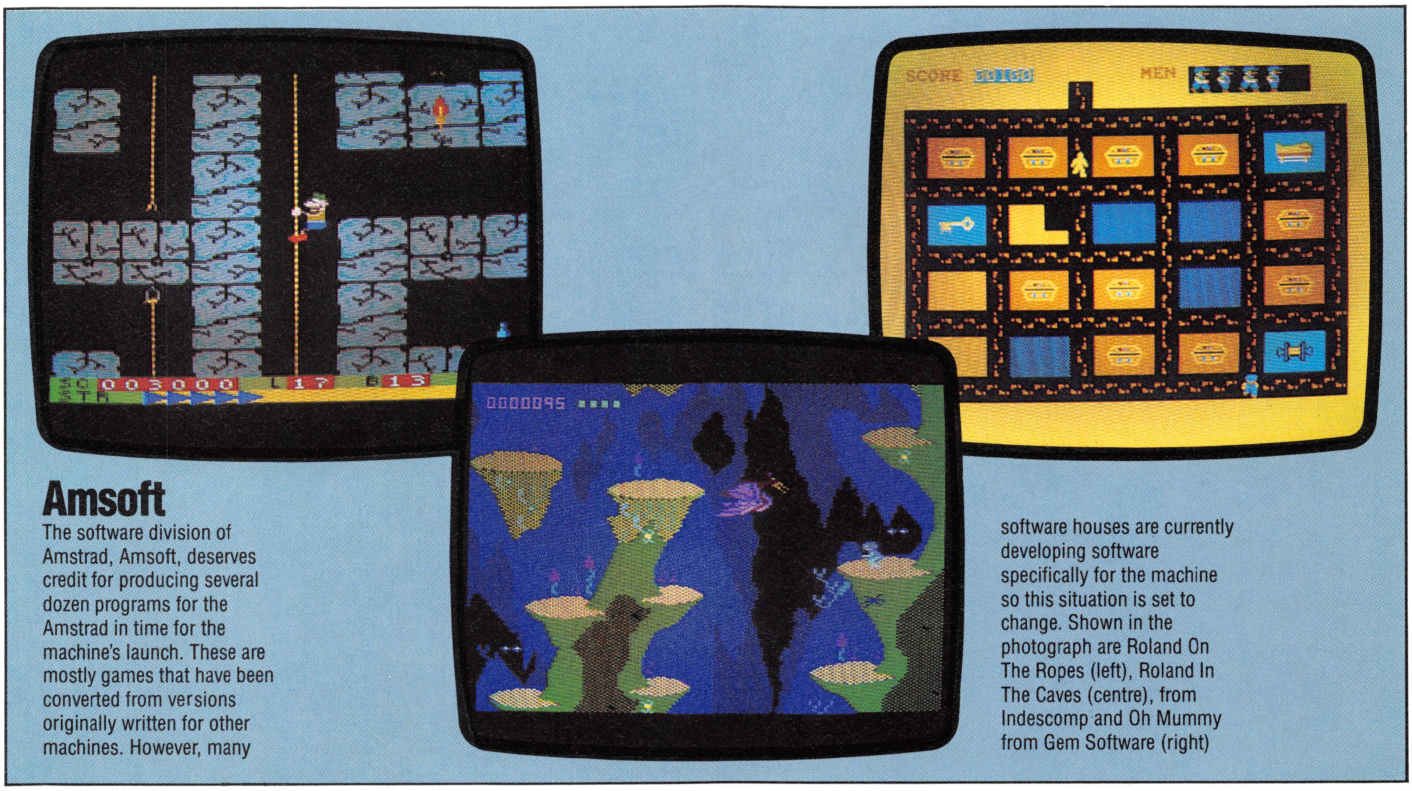
DOCUMENTATION
The beginner's guide included with the machine is easy to understand. BASIC reference manual and technical reference manual are also available

STRENGTHS
The monitor and built-in cassette deck make the Amstrad a complete off-the-shelf system. It has a large memory, excellent and versatile graphics, and sophisticated stereo sound

WEAKNESSES
The cassette deck is not very robust in construction. Graphics commands in BASIC are not (as yet) up to the potential of the hardware

Future Upgrade

Amstrad plans to make an impact on the lower end of the business computer market by launching a disk drive. This will cost £200, which includes the standard CP/M operating system needed by most business programs



Amsoft

The software division of Amstrad, Amsoft, deserves credit for producing several dozen programs for the Amstrad in time for the machine's launch. These are mostly games that have been converted from versions originally written for other machines. However, many software houses are currently developing software specifically for the machine so this situation is set to change. Shown in the photograph are Roland On The Ropes (left), Roland In The Caves (centre), from Indescomp and Oh Mummy from Gem Software (right)

CHRIS STEVENS

for BASIC programs and data.

The Amstrad BASIC is one of the most sophisticated versions of the language available. Extensive support is given for the excellent graphics hardware. There are several useful features to make the plotting of pictures onto the screen easier. The graphics origin may be redefined from the bottom left-hand corner of the screen to any point in or out of view. A graphics window can be set up to restrict graphics operations to a small part of the screen. As many as eight text windows may be defined on the screen at once, and text may be directed to whichever one is needed at the time. Prompts could be sent to one window, for example, and typed answers to another.

The BASIC graphics also lack a method of filling an area of the screen with a colour. There is no command to draw solid shapes on the screen and no command to fill in an outline already present — only lines and points may be drawn. The only way to produce a block of colour is to draw many lines close together, which is hardly an efficient method. However, there is a possibility that this omission will be rectified in the near future. Just as the BASIC ROM is switched in and out of the memory map to make room for the screen memory, so any other ROM may be added on to occupy the same space. The features missing from the current Amstrad BASIC could well appear on an add-on ROM and the new functions would then merge into the old BASIC. Complete new languages, such as PASCAL, FORTH and LOGO can be added in the same way. These 'sideways' ROMs would take up no more memory space than the current BASIC does, so 42 Kbytes of RAM would be available for their use as well. In fact, extra memory can be paged into the memory map in a similar way, so the standard 64 Kbytes of RAM can be expanded.

The most original aspect of Amstrad BASIC is its treatment of 'interrupts'. Many micros allow machine code programmers to use the interrupt system built into a machine's operating system to drive their own machine code routines. Amstrad BASIC takes this idea one stage further by allowing interrupts to be used from BASIC. The BASIC command AFTER will send a program to a specified subroutine after a certain period has elapsed. EVERY will do the same thing repeatedly. This advanced feature makes writing any kind of time-dependent program, from a laboratory data-gathering program to an arcade game, easier and more effective.

The Amstrad is unique among home computers in that it is supplied with a monitor instead of the television display used by its competitors. Monitors give better quality, so this is a definite advantage. However, users who have purchased the monochrome version may well find a need for a colour display occasionally. As it stands, there is no way to use a colour television set with the Amstrad, although an adaptor is available as an extra. It is possible to use a separate colour monitor with the Amstrad without the aid of the adaptor but, because the micro draws its power through the monochrome monitor, both monitors would need to be on at the same time.

With superb graphics and reliable hardware, an advanced BASIC and expansion potential, the Amstrad CPC 464 is one of the most sophisticated home computers now available. It is also excellent value for money.

JUNGLE FEVER

Ultimate, Play The Game made its name selling high-quality arcade-style games for the Sinclair Spectrum, and its recent releases have combined graphic excellence, fast action and adventure strategy. Atic Atac (see page 376) was the first of this new breed. Here, we look at Ultimate's latest chart-topping hit, Sabre Wulf.

As sales of arcade action games have started to fall off, software companies have turned their attention to a new form of game that combines elements of arcade strategy and adventure gaming. Ultimate, Play The Game's new release, Sabre Wulf, follows this formula.

An adventure game, in which the player must transport the hero — often a character from science fiction or fantasy literature — through various locations, solving puzzles on the way, often has laborious stretches. The action can cease for long periods of time, while the player attempts to find the answer to a seemingly insoluble riddle. The arcade game, on the other hand, does little to encourage prolonged thought, requiring instead good reactions and a quick trigger finger.

Sabre Wulf attempts to combine the best of both of these types of games. It is basically a maze game, set in jungle scenery, and is loosely derived from the arcade classic, Pac Man. The hero, an Indiana Jones look-alike, is guided through a highly complex maze, avoiding attackers and acquiring treasure to score points. The object of the exercise is to retrieve the four scattered pieces of a broken magic amulet.

The jungle scenery is superb, vividly portrayed in some of the most detailed graphics you will see on the Spectrum. Animals, plants, mountains, caves and treasure are all beautifully depicted, and at times the overall effect is reminiscent of a Rousseau painting. The maze is highly complex, and it is quite an effort to cover as much as 20 per cent of it in one game.

Most of the attacking nasties are simply dispatched with a quick flick of your hero's sword, though he must be facing them at the time. In this respect, Sabre Wulf mirrors Pac Man. But some opponents require special weapons, and these must be discovered as you journey through the maze. Other objects will give you an extra life — which is very important in this game, as only the best players are likely to remain unscathed for long. Some objects, such as the flowering orchids, can either help or hinder you. Depending on their colour, they can make you immune to danger, turn you temporarily into a vegetable, double the speed at which you move, or — most confusingly — reverse the effects of the controls. The effects of the orchids soon wear off.

Sabre Wulf is, in general, very well designed, although it suffers from some of the common faults that afflict other games software. The sound, initially highly entertaining and certainly complex, soon becomes annoying, and Ultimate has failed to include the facility to turn it off. Although supposedly either a one- or two-player game, it is in fact merely a one-player game that has two scoreboards. There is the usual Ultimate Hall of Fame, with space for the six highest scores. Here, Ultimate has opted for the arcade style of entering names: initials are entered by using the movement controls, on either the joystick or the keyboard.

Software manufacturers are at last beginning to recognise that there is a wide range of keyboards available for the Spectrum, and Ultimate has included provision for many of these. Keyboard use is less satisfactory, as Sabre Wulf employs the Q, W, E, R and T keys for movement and swordplay. This is difficult to understand, as all these keys fall in a single row and thus are extremely difficult to use.

But, minor quibbles about keyboard layout aside, Sabre Wulf is an excellent game, giving a fine balance between arcade action and adventure strategy. There is no doubt that it will prove popular for some time yet, and it is certainly a worthy successor to Ultimate's previous offerings.

Sabre Wulf: For 48K Spectrum, £9.95
Publishers: Ashby Computers and Graphics Ltd, Ashby de la Zouch, Leicestershire LE6 5JU
Authors: Ultimate, Play The Game
Joysticks: Kempston, Interface 2 and cursor
Format: Cassette

LIZ HEANEY

Title Page
This screen shows the graphics of the title page of Sabre Wulf. This same design is duplicated on the cover of the Sabre Wulf package.

Sabre Wulf is a maze game with an ingenious variety of nasty characters. The borders of the maze are filled with excellent graphics imitating a jungle scene

COLLISION COURSE

In the first two instalments of the project we designed routines to set up the scenario for our Minefield game. Now, we look at the control of movement from the keyboard, and inspect the parts of the program that detect collisions between the player characters and the mines.

BBC BASIC has no fewer than four commands that respond to a single keypress. The choice of command will obviously depend on the desired usage. INKEY$ and INKEY are normally used when you want to wait a certain length of time for a possible keypress before carrying on with the rest of the program; GET$ and GET on the other hand will always halt program execution until a key is pressed. These last two commands tend to be used when a response to a question is required, such as 'Another Game Y/N?'. If GET or GET$ is used then the program will wait for an answer. In this case, the only acceptable answers are 'Y' and 'N'. We can use a loop to REPEAT the GET instruction UNTIL the answer is 'Y' or 'N', as follows:

```
1000 PRINT "ANOTHER GAME Y/N?"
1010 REPEAT
1020    A$=GET$
1030 UNTIL A$="Y"OR A$="N"
1040 Etc
```

If GET$ or INKEY$ is used then the key pressed is interpreted as its character string, as in the above example. If we use GET or INKEY then a numeric rather than a string value is returned; this value is the ASCII code of the key pressed. These options allow the programmer to test for keys that do not have a corresponding character, such as the Return or cursor keys. The statement *FX4,1 may be used to make the cursor keys return ASCII codes. If this is done, the keys have the values:

Left cursor	136
Right cursor	137
Down cursor	138
Up cursor	139

Let us say that we wished to accept only left and right cursor inputs to our program. The following program segment uses INKEY to wait for a quarter of a second for an input:

```
1000 *FX4,1:REM TURN ON CURSOR ASCII MODE
1010 REPEAT
1020    A=INKEY(25)
1030 UNTIL A=136 OR A=137
1040 *FX4,0:REM RESTORE CURSOR TO EDIT MODE
```

The parameter 25 in line 1020 tells the computer to wait 25 hundredths of a second before going on with the program.

These statements do not test the keyboard itself but affect an area of memory inside the computer called the keyboard buffer. This is a temporary storage space for characters that are input from the keyboard, and is rather like a cinema queue. New characters typed in tag on to the end of the queue and the processor takes characters from the front of the queue. In this way, if you type in characters faster than the processor can handle them, they are not lost but just wait their turn in the keyboard buffer. As INKEY, GET, INKEY$ and GET$ normally test the front of the keyboard buffer queue you have no way of knowing how long a particular character has been sitting in the buffer waiting to be processed. In games that are controlled from the keyboard this can make for a sluggish response, as the program may be processing earlier key presses whilst the player is making new ones. For example, if you fill the keyboard buffer with cursor-right codes then all of these will need processing before the program can respond to a cursor-left command. This can leave the player frantically pressing the cursor-left button and wondering why the object being controlled is still moving right!

There are two solutions to this problem. The first is always to clear out the keyboard buffer just prior to testing it. This can be done using the statement *FX15,1. Alternatively, we can use a further variation of INKEY. As described above, INKEY( ) waits for a length of time, specified by the number in the bracket, for a key to be pressed before continuing with the program. We can, however, make INKEY test the keyboard instead of the keyboard buffer by specifying a *negative* number in the brackets following the command. Each key has a negative number assigned to it for this purpose, a full list of which is given on page 275 of the User Guide. In our program, we shall use the cursor keys to control movement. The values of the keys to be used with INKEY are:

Cursor-left	-26
Cursor-right	-122
Cursor-down	-42
Cursor-up	-58

The following procedure uses INKEY to test the keyboard directly for each of the four cursor keys in turn. If one of the keys is being pressed then another procedure ('move') is accessed, with two parameters being passed. These parameters hold information about the direction in which the mine

detector character is to be moved.

```
3000DEF PROCtest_keyboard
3010 REM ** UP ? **
3020IF INKEY(-58)=-1 THEN PROCmove(0,-1)
3030REM ** DOWN ? **
3040IF INKEY(-42)=-1 THEN PROCmove(0,1)
3050REM ** RIGHT ? **
3060IF INKEY(-122)=-1 THEN PROCmove(1,0)
3070REM ** LEFT ? **
3080IF INKEY(-26)=-1 THEN PROCmove(-1,0)
3090ENDPROC
```

THE PROCEDURE 'MOVE'

This procedure is central to the program. Within it the detector and assistant characters are moved, and collisions with mines are tested for. Let us first look at the section of the procedure that controls the movement of the characters.

Two parameters are passed into 'move' from the procedure 'test-keyboard'. These are accepted into the variables delta-x and delta-y for use within 'move', and correspond to the change to be made to the x and y co-ordinates of the mine detector. For example, if the cursor-up key were pressed then the values 0 and −1 would be passed to 'move'. The instructions: xdet = xdet+delta-x and ydet = ydet+delta-y cause the co-ordinates of the detector to be updated. In the case of cursor-up, 0 is added to xdet and −1 is added to ydet, effectively subtracting one from its value. This seems to imply a move of one unit *down* the screen, but we must remember that the origin for character positions is the top left corner and y values increase down the screen. Thus, reducing ydet by one causes an upward movement of one character cell. You may wish to verify that the values passed for the other three directions do in fact correspond to the correct alterations of xdet and ydet. It would be quite feasible to use this system to include diagonal movements. Passing the values (1,-1) to 'move' would cause the detector to move diagonally one cell up and one cell right. However, other keys would have to be introduced at this stage to allow diagonal control from the keyboard.

Here is the listing for the 'move' procedure:

```
3220 DEF PROCmove(delta_x,delta_y)
3230REM ** RUB OUT OLD POSITIONS **
3240COLOUR 1
3250PRINTTAB(xdet,ydet);" "
3260PRINTTAB(xman,yman);" "
3270REM ** MOVE DETECTOR **
3280xdet=xdet+delta_x
3290ydet=ydet+delta_y
3300REM ** TEST FOR LIMITS **
3310IF xdet>17 THEN xdet=17
3320IF ydet>25 THEN ydet=25
3330IF xdet<2  THEN xdet=2
3340IF ydet<1  THEN ydet=1
3350REM ** CALCULATE MAN'S COORDS **
3360xman=19-xdet
3370yman=26-ydet
3380PROCconvert(xman,yman)
3390IF POINT(xgraph,ygraph)=2 THEN PROCexplode(xgraph,ygraph)
3400PROCconvert(xdet,ydet)
3410IF POINT(xgraph,ygraph)=2 THEN PROCfound_mine
3420PROCposition_chars
3430ENDPROC
```

Before the x and y co-ordinates of the detector are altered we must first erase the old positions of the detector and the assistant. Lines 3250 and 3260 use the old values of xdet, ydet, xman, and yman to PRINT spaces over the old characters. As the new characters will be PRINTed in red (logical colour 1) the colour command is used in line 3240 to set the current foreground colour to 1. Lines 3280 and 3290 update the co-ordinates of the detector as described previously. Before actually PRINTing the detector in its new position, tests must be made to ensure that we are not incrementing or decrementing co-ordinates outside the area we have defined as our minefield. The upper and lower limits of xdet and ydet are tested in lines 3310 to 3340. Here it has been decided that if the detector reaches a boundary then it will stay there until moved in the opposite direction. For example, line 3310 tests to see if the right-hand edge of the minefield has been reached, denoted by an x co-ordinate of 17. If an attempt is made to increase xdet past 17, then this line simply resets the value to 17. It would have been equally possible to create a 'wrap-around' effect in which the detector, on reaching the right-hand boundary, would next appear back on the left-hand side of the screen. To produce a wrap-around effect at the right-hand edge of the minefield, we alter line 3310 to read:

```
3310 IF xdet>17 THEN xdet=2
```

You may wish to alter this and the other three boundary tests to provide wrap-around on all edges of the minefield.

One of the rules of our game is that while the player moves the mine detector around the minefield destroying mines the player's assistant mirrors every move. In order to do this, we must automatically update the assistant's co-ordinates, which are related to the detector's co-ordinates by a simple pair of formulae as shown in lines 3360 and 3370. To demonstrate how these produce mirror movements let us look at the relationship between the x co-ordinates (xman=19−xdet).

Initially, xdet is 2 and xman is 17. If the detector moves one place to the right, xdet will increase to 3. Using the formula above, xman will be calculated as 19−3 = 16. This means that the assistant moves one place to the left. If xdet moves right again, then xdet will become 4 and xman will be 15, and so on. The y co-ordinates operate similarly.

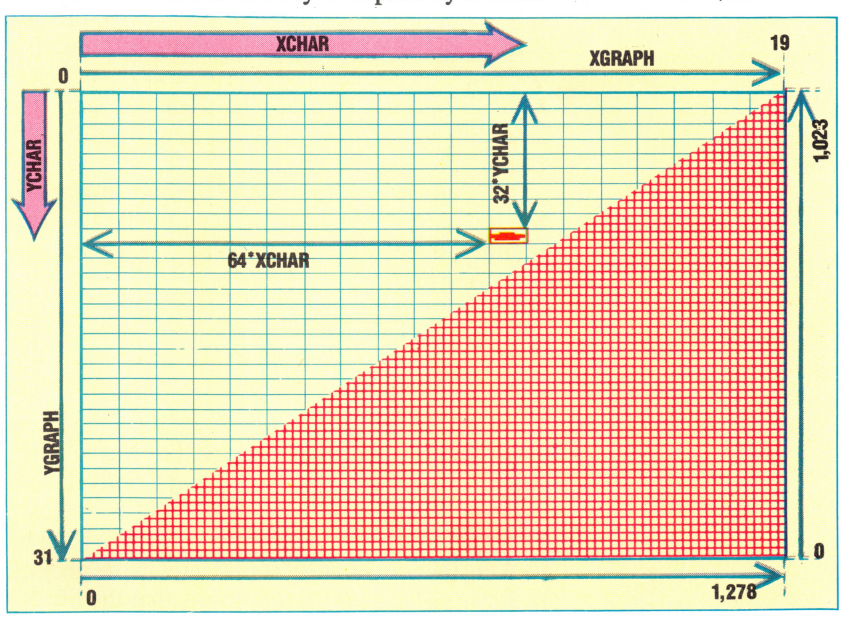
Before we PRINT the detector and the assistant we have one remaining task. We must check to see if either the assistant or detector is moving into a character cell that is already occupied by a mine.

Mapping

The game mixes high resolution graphics with the BBC/Electron text display. This has its advantages but means two different co-ordinate systems must be mixed, one for graphics and one for text. The BBC/Electron has several different text formats and so each has its own co-ordinate system. The game uses mode five, which gives 20 characters across the screen and 32 down. This is shown in the top and left-hand part of the diagram.

The machines also have three different graphics resolutions, but at least these all use the same co-ordinate system. This treats all modes as having a resolution of 1,280 by 1,024. This is shown in the bottom and right-hand part of the diagram.

The program uses the high resolution co-ordinate system to read points off the low resolution text display. This means converting a character position to a high resolution co-ordinate. To do this the horizontal co-ordinate (XCHAR in the program) must be multiplied by 64 and the vertical co-ordinate (YCHAR) by 32. One other problem has to be overcome. The text screen co-ordinates start with zero at the top of the screen and count down, while the graphics co-ordinates start with zero at the bottom and count up. This is easily solved by subtracting 32*YCHAR from 1,023



LIZ DIXON

BBC BASIC allows us to test any point on the screen to see if that point is lit in a certain colour. POINT(X,Y) will return the colour of the pixel at position (X,Y). We can use this to see if the colour of the cell we are about to move into is green (i.e. it contains a mine). There is only one snag. POINT(X,Y) uses the high-resolution co-ordinate system to specify the point to be looked at. If we want to use this command for our game we must first convert our character cell co-ordinates into grahics co-ordinates. The easiest point in the cell to specify would be the centre.

In the last section we worked out that in mode 5 each character cell is 64 graphics units wide and 32 graphics units high (see page 404). Multiplying xchar by 64 will give the xgraph co-ordinate of the edge of the cell in question. Adding a further 32 to xgraph will give the x co-ordinate of the centre of the cell. Calculation of ygraph is rather more complex as the two systems run in opposite directions. At the top of the screen ygraph is 1023. Working down, 32*ychar would bring us to the top of the specified cell; moving down a further 16 units would bring us to the y co-ordinate of the centre of the cell. The following procedure can, therefore, be designed to convert character co-ordinates to graphics co-ordinates:

```
3720 DEF PROCconvert(xchar,ychar)
3730xgraph=64*xchar+32
3740ygraph=1023-(32*ychar+16)
3750ENDPROC
```

We can see the true value of being able to pass parameters between procedures if we look back to the procedure 'move'. The 'convert' procedure is used twice: first of all, to convert the co-ordinates of the assistant into graphics co-ordinates, and these values are then used in line 3390 to test for the colour green. (Remember that although the assistant's co-ordinates have been updated, the character has not yet been PRINTed in its new position.) If the colour present is green then the program jumps to another procedure to display an explosion. The 'convert' procedure is used for the second time in line 3400, but in this instance the character co-ordinates of the detector are calculated. Line 3410 then tests to see if the cell is occupied by a mine. If it is, the procedure 'found-mine' is called. Finally, the detector and the assistant are PRINTed in their new positions by calling 'position-chars'.

We shall be looking at the procedure 'explode' in the next instalment, and so for now let us put a dummy procedure in its place. Type in the following lines:

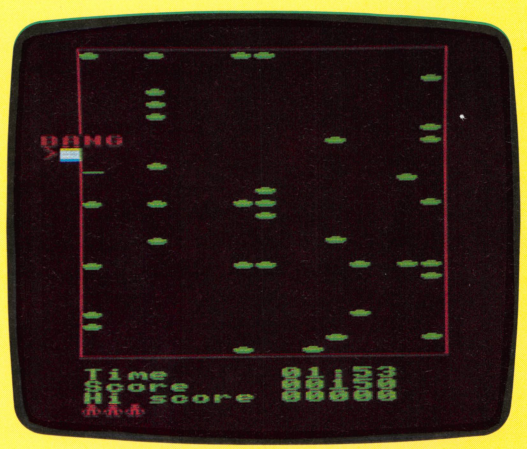
```
3550 DEFPROCexplode(x-explode,y-explode)
3560 PRINT"BANG"
3570 END
3580 ENDPROC
```

Let's finally look at the procedure 'found-mine', called when the detector moves into a cell occupied by a mine. A sound effect to indicate finding a mine would be a nice idea. We shall be looking in more detail at sound later in the project, so for now all we need to know is that the SOUND statement in line 3790 produces a high-pitched 'ping'. The main function of this routine, however, is to increment the player's score. The program uses two variables for the score, the first of which is a numeric variable that is incremented by 150. So that the score is always PRINTed as a five-digit number, we must add leading zeros to its numeric values. In order to do this, we must first convert the numeric value of the score to a string variable and then use string-handling techniques, as described earlier in the project (see page 404) to add on leading zeros. The complete procedure is:

```
3770DEF PROCfound_mine
3780REM ** SOUND EFFECT **
3790SOUND 2,-15,170,3
3800REM ** INCREMENT SCORE **
3810COLOUR 2
3820score=score+150
3830score$=STR$(score)
3840 score$=LEFT$(zero$,5-LEN(score$))+score$
3850PRINTTAB(11,28);score$
3860ENDPROC
```

In the last part of the project, we wrote a short calling program for the procedures we have written so far. The procedures we have given here can be added to your program with the line numbers shown. The only alteration we need to make, at this stage, to get the program to run is to call the procedure 'test-keyboard' from within the main loop of the calling program. Therefore, you will need to add the following temporary line to your program:

```
55 PROCtest-keyboard
```



IAN McKINNELL/LIZ HEANEY

"BANG" On Target
At this point in the Minefield project, the program will fill the screen with randomly placed mines; create your character and a mirror image; define score tables, and provide movement. Because the program is not complete, if you run into a mine you will only see the word "BANG" printed on the screen. In the next instalment of this project, we will design a routine that creates an actual explosion with attendant sound effects. It is also possible that you may encounter error messages at particular points in the game's execution. These messages arise because of the incomplete structure of the game, and will be cleared up as the final subroutines are added in the next instalments. Still, we have reached a point where the program has assumed the characteristics of a fast-action game

REFLEX ACTION

It soon becomes painfully obvious to most of us that our reflexes become slower as we get older — which is why children and teenagers perform better than adults in arcade games, where a quick response is all-important. The program presented here allows you to measure your reaction times by playing a simple game.

We all like to think that our reflexes are pretty good; in fact we tend to assume that our reactions to events are near enough instantaneous. However, our quickest response to a stimulus is likely to be around one-third of a second, which seems fast enough until you consider that a speeding car will have travelled around 10 metres (over 30 feet) in this time. Reaction timings are rarely consistent — alcoholic excess, tiredness or illness can all have an adverse effect on results. Our program is designed to allow anyone's reflexes to be tested, and to make things fairer, the timings are averaged out over five trials.

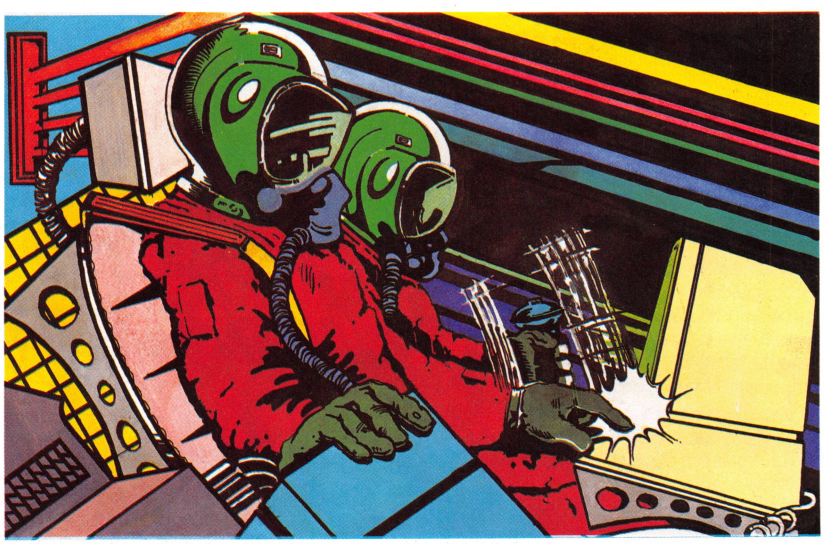
You play the role of a spaceship pilot who must deliver urgent medical supplies to a colony in the asteroid belt. Speed is vital if the supplies are to arrive in time, but the ship must stop as soon as a collision is imminent. Simply pressing the space bar will stop your craft, allowing you to steer your way past the asteroids. The game displays the time taken between the asteroids appearing on screen and the space bar being pressed. Once you have stopped the ship five times, your average reaction time is calculated. Just to make sure that you don't cheat, the program checks to see if the space bar is held down continuously — if this is the case your engines will stop and a warning noise will sound. At the same time as the asteroids appear on the screen, your ship's radar will issue a high-pitched note. As an exercise, try changing the program so that you receive either an audible or a visual warning, but not both.

A final modification could be to select from two or more options presented on the screen. This would make the program more of a test of judgement and less an indication of simple reaction times.

Testing Time

```
 10 REM *FOR THE SPECTRUM
 20 INK 7: PAPER 0: BORDER 0
 30 FOR T=0 TO 31: READ A
 40 POKE USR "A"+T,A: NEXT T
 60 DATA 123,231,255,255,245,233,123,100
 70 DATA 34,187,208,49,86,96,178,205
 80 DATA 67,234,45,123,198,255,29,245
 90 DATA 78,100,245,60,50,160,245,189
150 LET M=0: LET A=0: LET V=0: DIM R(5): CLS
220 FOR G=1 TO 5
250 FOR P=1 TO RND*150+50
260 PLOT INT (RND*250),INT (RND*170)
270 IF INKEY$<>" " THEN  GO TO 280
275 BEEP .1,-3: GO TO 265
280 NEXT P
290 FOR A=144 TO 147
320 PRINT AT RND*20,RND*30;CHR$ (A)
330 NEXT A
350 BEEP .05,15: LET C=0
360 LET C=C+1: LET A$=INKEY$
410 IF A$<>" " THEN  GO TO 355
420 LET R(G)=C/66+.05
430 PRINT AT 19,0;"YOU TOOK ";INT (R(G)*100)/100;
440 PRINT " SECONDS TO STOP  "
445 PRINT "THE ENGINES "
450  LET M=M+R(G)
453 FOR J=1 TO 300: NEXT J
455 NEXT G
460 LET A=M/5: CLS
480 PRINT "OVER FIVE TURNS YOUR AVERAGE "
490 PRINT "REACTION TIME WAS ";INT (A*100)/100;
495 PRINT " SECS  "
500 FOR G=1 TO 5: LET V=V+ABS (R(G)-A)
510 NEXT G
520 PRINT : PRINT "YOUR REACTION TIME VARIED BY"
530 PRINT INT (V*20/A);"  PER CENT"
540 PRINT : PRINT "DO YOU WANT ANOTHER GO? (Y/N)"
550 LET R$=INKEY$: IF R$="y" THEN  GO TO 150
560 IF R$<>"n" THEN  GO TO 550
570 INK 0: PAPER 7: BORDER 7: CLS
```

```
 10 REM *FOR THE BBC MICRO AND ELECTRON
 20 MODE 1:CLS:VDU 23;8202;0;0;0:DIM REACT(5)
 30 REM *SET UP CHARACTERS (ASTEROIDS)
 40 FOR J=0 TO 3
 50 VDU 23,237+J,78,100,245,60,50,160,245,189
 60 VDU 23,233+J,89,67,34,156,123,200,256,29
 70 NEXT J
 80 SUM=0:AVERAGE=0:VARY=0:CLS
 90 FOR GO=1 TO 5
100 REM *PLOT STARS
110 FOR DELAY=1 TO 100+RND(300)
120 PLOT 69, RND(1200),RND(1012)
130 IF INKEY(-99)=0THEN 150 ELSE
140 SOUND 1,-15,5,1:GOTO 130
150 NEXT DELAY
160 REM *PLOT ASTEROIDS AROUND SCREEN
170 FOR ASTEROID =233 TO 240
180 X=RND(28):Y=RND(28)
190 PRINT TAB(X,Y);CHR$(ASTEROID)
200 NEXT ASTEROID
210 SOUND 2,-15,150,3
220 REM * TEST SPACE BAR
230 TIME=0
240 IF INKEY(-99)=0 THEN 240
250 REACT(GO)=TIME/100
260 PRINT TAB(3,26);"YOU TOOK ";
270 PRINT;REACT(GO);" SECONDS   "
280 PRINT TAB(3,28);"TO STOP THE ENGINES"
290 FOR I=1 TO 500:NEXT I
300 SUM=SUM+REACT(GO)
310 NEXT GO
320 FORI=1 TO 3000:NEXT I
330 REM *CALCULATE RESPONSE TIME
340 AVERAGE=SUM/5
350 FOR GO=1 TO 5
360 VARY=VARY+ABS(REACT(GO)-AVERAGE)
370 NEXT GO
380 CLS:PRINT:PRINT
390 PRINT "OVER FIVE TURNS YOUR AVERAGE"
400 PRINT"REACTION TIME WAS ";AVERAGE;" SECS."
410 PRINT:PRINT"YOUR REACTION TIME VARIED BY"
420 PRINT INT(VARY*20/AVERAGE);" PER CENT"
430PRINT:PRINT"DO YOU WANT ANOTHER GO Y/N ?"
440 R$=INKEY$(0):IF R$="Y" THEN 80
450 IF R$<>"N" THEN 440 ELSE MODE 4
```



MIKE CLOEWS

CIRCLE OF LIGHT

BBC BASIC is graced with a range of commands to help create impressive graphics. Yet it lacks one important facility — a command that enables you to draw circles. Here, we show you how to write a machine code routine to draw circles, which can be used from BASIC or incorporated as a routine in other programs.

Drawing a circle may seem like a simple task, but coming up with a mathematical equation to plot the points on a circle with speed is a fairly difficult exercise. The simplest way to draw a circle is by using the COS and SIN functions, as in this example:

```
DEF PROCCIRCLE (XORG,YORG,R)
  MOVE XORG+R, YORG
  FOR THETA =0 TO 2*PI STEP PI/32
    X=R*COS(THETA)
    Y=R*SIN(THETA)
    DRAW X+XORG,Y+YORG
  NEXT THETA
ENDPROC
```

Because this procedure uses the SIN and COS functions it takes some time to perform the calculations. The plotting of the circle is relatively slow. However, it can be speeded up considerably using the following algorithm, which can be derived from elementary geometry and differential calculus:

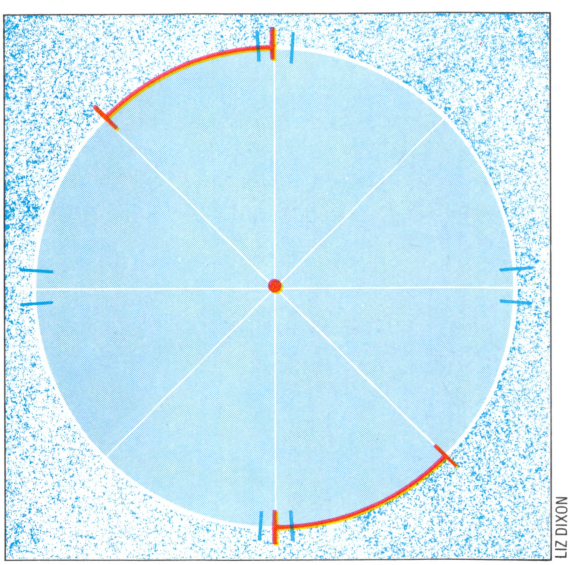
```
  3 MODE1
  5 PROCCIRCLE(500,600,200)
  7 END
 10 DEF PROCCIRCLE (XORG,YORG,R)
 20 Y=R
 30 FOR X=1 TO Y*.707
 40 Y=Y-X/Y
 50 PROCPOINTS(X,Y)
 60 NEXT
 70 ENDPROC
 80 DEF PROCPOINTS (X,Y)
 90 PLOT 69,XORG+X,YORG+Y
100 PLOT 69,XORG-X,YORG+Y
110 PLOT 69,XORG-X,YORG-Y
120 PLOT 69,XORG+X,YORG-Y
130 PLOT 69,XORG+Y,YORG+X
140 PLOT 69,XORG-Y,YORG+X
150 PLOT 69,XORG-Y,YORG-X
160 PLOT 69,XORG+Y,YORG-X
170 ENDPROC
```

This routine draws the circle in eight sections at the same time, which helps make the plotting much quicker. This algorithm is also better than our initial routine because it doesn't require sine and cosine values to be calculated for each point. However, it does need to calculate a division for each point, which is slow to perform.

An alternative that doesn't need any complicated maths is:

```
 10 MODE4
 20 PNUM=69
 30 PROCCIRCLE(500,600,200)
 40 END
 50 :
 60 DEF PROCCIRCLE(X,Y,R)
 70 VDU29,X;Y;:REM SET GRAPHICS ORIGIN
 80 X=0:Y=R:D=3-2*R:REM VARIABLES
 90 REPEAT
100 PROCCPLOT
110 IFD<0:D=D+4*X+6:ELSED=D+4*(X-Y)+10:Y
    =Y-4
120 X=X+4
130 UNTIL X>Y:ENDPROC
140 :
150 DEF PROCCPLOT
160 PLOT PNUM,X,Y
170 PLOT PNUM,Y,X
180 PLOT PNUM,Y,-X
190 PLOT PNUM,-X,Y
200 PLOT PNUM,-X,-Y
210 PLOT PNUM,-Y,-X
220 PLOT PNUM,-Y,X
230 PLOT PNUM,X,-Y
240 ENDPROC
```

This is known as Breshen's algorithm. It is much quicker because it only has to perform addition, subtraction, and multiplication by two or four (both of which can be implemented by bit shifting). This is the algorithm that we use in our machine code circle drawing program.

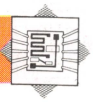


LIZ DIXON

Mirror Images

The machine code program uses an equation to plot the circle that is specially chosen to be fast.. All the same it would be slow to use it to draw all the way around a circle, calculating each point. This is because the top half of a circle is a mirror image of the bottom half, so we only need calculate the equation for the top half and as we do this produce its mirror image below. This same principle can be extended because the left half of a circle is a mirror image of the right half.

In practice, the program only calculates one eighth of a circle. Each point on this eighth is copied seven times onto other parts of the circle to build up the whole circle



BBC Circles

```
 30PNUM=69
 40S%=2
 500SWRCH=&FFEE
 60DIM CODE% 600
 70DIM D% 12
 80X=D%+2
 90Y=D%+4
100D=D%+6
110N1=D%+8
120N2=D%+10
130PROCCOMPILE
140MODE4
150PROC_OLYMPIC
160END
170#
180DEF PROCCOMPILE
190FOR I% =0 TO S% STEP S%
200K%=P%
210P%=CODE%
220[OPT I%
230.CIRCLE
240JSR INIT
250:
260.LOOP
270JSR COMPXY:BMI DOIT:JSR CPLOT:RTS
280.DOIT
290JSR CPLOT
300LDA D+1:BPL D_IS_POS
310:
320.D_IS_NEG
330JSR DNEG
340JSR ADD_4_TO_X
350JMP LOOP
360:
370.D_IS_POS
380JSR DPOS
390JSR ADD_4_TO_X
400JMP LOOP
410:
420.INIT
430LDY #8
440.L7
450LDA &601,Y:STA &80,Y
460DEY:BPL L7
470INY
480LDA (&80),Y:STA X
490LDA (&83),Y:STA Y
500LDA (&86),Y:STA D
510INY
520LDA (&80),Y:STA X+1
530LDA (&83),Y:STA Y+1
540LDA (&86),Y:STA D+1
550LDA #29:STA D%+1
560LDA #0:STA D%
570JSR PSTR
580LDA #25:STA D%
590LDA #PNUM:STA D%+1
600JSR SETD
610RTS
620:
630.COMPXY
640LDA X:STA N1:LDA X+1:STA N1+1
650LDA Y:STA N2:LDA Y+1:STA N2+1
660JSR SUB
670LDA N1+1
680RTS
690:
700.CPLOT
710LDX #4
720.L2
730JSR P2
740DEX:BNE L2
750RTS
760:
770.DNEG
780LDA X:STA N1:LDA X+1:STA N1+1
790JSR TIMES4
800LDA #6:STA N2:LDA #0:STA N2+1
810JSR ADD
820LDA D:STA N2:LDA D+1:STA N2+1
830JSR ADD
840LDA N1:STA D:LDA N1+1:STA D+1
850RTS
860:
870.DPOS
880LDA X:STA N1:LDA X+1:STA N1+1
890LDA Y:STA N2:LDA Y+1:STA N2+1
900JSR SUB
910JSR TIMES4
920LDA #10:STA N2:LDA #0:STA N2+1
930JSR ADD
940LDA D:STA N2:LDA D+1:STA N2+1
950JSR ADD
960LDA N1:STA D:LDA N1+1:STA D+1
970JSR SUB_4_FROM_Y
980RTS
990:
1000.ADD_4_TO_X
1010LDA #4:STA N1:LDA #0:STA N1+1
1020LDA X:STA N2:LDA X+1:STA N2+1
1030JSR ADD
1040LDA N1:STA X:LDA N1+1:STA X+1
1050RTS
1060:
1070.SUB_4_FROM_Y
1080LDA#4:STA N2:LDA #0:STA N2+1
1090LDA Y:STA N1:LDA Y+1:STA N1+1
1100JSR SUB
1110LDA N1:STA Y:LDA N1+1:STA Y+1
1120RTS
1130.SETD
1140LDA #0:STA X:STA X+1
1150LDA D:STA Y:LDA D+1:STA Y+1
1160ASL D:ROL D+1
1170LDA #3:STA N1:LDA #0:STA N1+1
1180LDA D:STA N2:LDA D+1:STA N2+1
1190JSR SUB
1200LDA N1:STA D:LDA N1+1:STA D+1
1210RTS
1220:
1230.P2
1240JSR PSTR
1250JSR SWAPXY
1260JSR PSTR
1270JSR NEGY
1280RTS
1290:
1300.TIMES4
1310ASL N1:ROL N1+1
1320ASL N1:ROL N1+1
1330RTS
1340:
1350.ADD
1360CLC
1370LDA N1:ADC N2:STA N1
1380LDA N1+1:ADC N2+1:STA N1+1
1390RTS
1400:
1410.SUB:\ (N1=N1-N2)
1420SEC
1430LDA N1:SBC N2:STA N1
1440LDA N1+1:SBC N2+1:STA N1+1
1450RTS
1460:
1470.PSTR
1480LDY #250
1490.L1
1500LDA D%-250,Y
1510JSR OSWRCH
1520INY
1530BNE L1
1540RTS
1550:
1560.SWAPXY
1570LDA X:PHA:LDA X+1:PHA
1580LDA Y:STA X:LDA Y+1:STA X+1
1590PLA:STA Y+1:PLA:STA Y
1600RTS
1610:
1620.NEGY
1630LDA #0:STA N1:STA N1+1
1640LDA Y:STA N2
1650LDA Y+1:STA N2+1
1660JSR SUB
1670LDA N1:STA Y
1680LDA N1+1:STA Y+1
1690RTS
1700]
1710NEXT
1720ENDPROC
1730#
1740DEF PROCCIRCLE(P1%,P2%,P3%)
1750CALL CIRCLE,P1%,P2%,P3%
1760ENDPROC
1770#
1780DEF PROC_OLYMPIC
1790PROCCIRCLE(300,600,150)
1800PROCCIRCLE(650,600,150)
1810PROCCIRCLE(1000,600,150)
1820PROCCIRCLE(475,450,150)
1830PROCCIRCLE(825,450,150)
1840VDU29,0;0;
1850MOVE100,250
1860DRAW100,800
1870DRAW1200,800
1880DRAW1200,250
1890DRAW100,250
1900ENDPROC
```

This routine makes it easy to draw circles on the BBC and Electron. Using the machine code routine is simply a matter of placing the required values in three INTEGER variables (e.g. X%, Y% and R%) and calling the routine with the command: CALL CIRCLE,X%,Y%,R%. This will draw a circle of radius R%, with its centre at X%,Y%. Notice that the graphics origin is moved to the centre of the circle by the routine. It can, however, be reset with VDU29,0;0;. Unfortunately, the CALL statement does not allow expressions to be used as parameters, and so X%, Y% and R% can only be variables. To overcome this difficulty we use the procedure PROCCIRCLE (as shown in the listing)

In order to enable the program to work in all graphics modes, we have used the VDU 25 command to plot all the points on the circle, bearing in mind that this does slow the program down considerably. The program is structured in self-contained subroutines to help aid understanding. Only simple straightforward techniques have been used to avoid confusion. The cost of this clarity has been a loss of speed in program execution. Nevertheless, to draw a circle with a radius of 300 units, the program executes in machine code in only 0.52 seconds, compared with 1.9 seconds for our original BASIC version.

Useful improvements can be made to the routine by altering the value of PNUM in line 30 from 69 to 5. This will make the program plot lines instead of points and so will draw a solid disk of colour rather than a circle. A side effect of making this change is that an unwanted line will be drawn to each circle. To get rid of this, add the instruction:

```
1745 VDU29,0;0;:MOVE P1%,P2%n
```

This can be done in assembler by adding the instructions:

```
575 LDA#25:JSR OSWRCH:LDA#4:JSR
    OSWRCH:LDA#0:JSR OSWRCH:LDA#0:JSR
    OSWRCH:LDA#0:JSROSWRCH:LDA#0:JSR
    OSWRCH
```

More complicated improvements could be made to the program to get it to draw arcs and ellipses.

LONE STAR

TI President
The President and Chief Executive Officer of Texas Instruments, Mr J Fred Bucy. Mr Bucy has occupied the position of President since April, 1976

Texas Instruments could be credited with creating the home computer revolution. In 1958, one of the company's engineers invented the integrated circuit, which is the basis of the microcomputer. Twenty years later, the TI99/4A — the first micro aimed specifically at the home user — was marketed by the company.

Introduced in 1978, the TI99/4A home computer used the Texas Instruments' 16-bit TMS9900 microprocessor, and had 16 Kbytes of RAM memory. There was a choice of 16 colours and three sound channels on the machine. Despite healthy sales, however, the computer fell victim to the ferocious price war that developed in the United States in 1982 and 1983. In the United Kingdom, the machine originally sold for around £1000 (£645 for the machine and the remainder for an American standard colour television). This price was progressively reduced to around £80. At the end of 1983, TI ceased production of the machine entirely.

Richard Mann, TI's European Public Relations Manager, says: 'The TI99/4A was selling extremely well, but it was not profitable and we lost several hundred million dollars.' Asked if TI intends to launch another machine, he replied: 'Absolutely not. We made that very clear at the time. We are more interested in selling machines for hundreds of dollars rather than tens of dollars.' However, TI intends to remain 'firmly in the professional computer market'.

The company's other products range from children's electronic toys, such as Speak and Spell, to minicomputers and sophisticated seismic detection equipment. It has 15 manufacturing plants around the world and an annual turnover of $4 billion.

Texas Instruments was founded by two scientists, John Karchner and Eugen McDermott, in 1930, and was originally named Geophysical Service Incorporated. Karchner had been investigating the idea of bouncing sound waves off geological strata to calculate their depth, and the company was set up in Dallas, Texas to sell this idea to the oil industry.

GSI grew steadily throughout the 1930s, developing new techniques and equipment for seismological surveying, and during the Second World War its surveying equipment proved useful in detecting submarines, which led to the establishment of a laboratory and manufacturing division of GSI. By 1951, this offshoot of the parent company had grown to such an extent that it was decided to establish it in its own right. The new company was christened Texas Instruments.

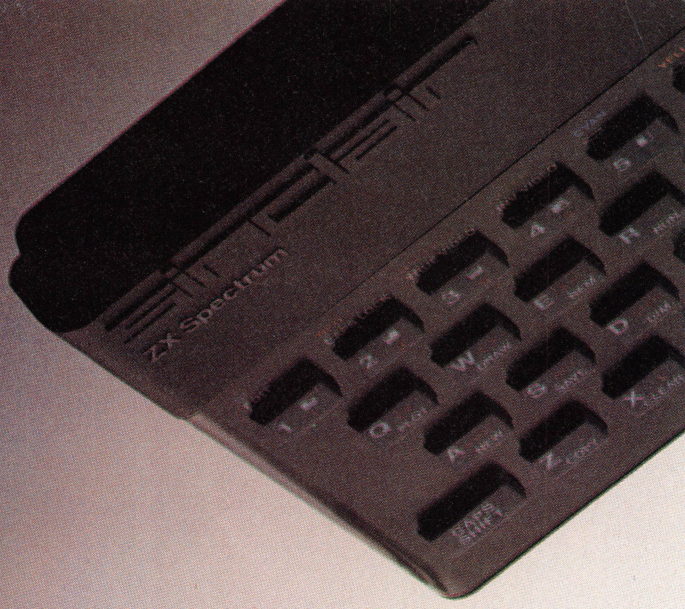
The following year, TI obtained a licence from Bell Laboratories to manufacture the newly-invented transistors, and in 1954 it produced the world's first transistor radio. In 1958, Jack Kilby, an engineer with the company, invented the integrated circuit, and TI has remained at the forefront of research in this field. Kilby also assisted in the invention of the world's first hand-held electronic calculator in 1967.

The TI Professional computer, which was launched in January 1983, is based on the 8088 processor, and can therefore run CPM-86 and MS-DOS. Later that year, TI introduced the Texas Instruments portable business computer and the hand-held CC40 micro.

Today, the company also produces a wide variety of electronic components including 64 Kbyte RAM chips, of which TI claims to be one of the world's largest suppliers. TI chips appear in almost every home computer, including the Commodore 64 and the Sinclair Spectrum. In common with most other chip manufacturers, TI has developed a range of 16-bit processors, including the TMS9900 featured in the TI99/4A. This processor was unusual in having no internal registers, which were included in a special 'scratchpad memory' external to the CPU. Richard Mann says: 'The TMS9900 was a bit ahead of its time. It had many powerful features, but the architecture was not seized upon.' Texas Instruments has now marketed a 16-bit processor known as the TMS99000, and this, the company hopes, will be more successful.

Texas Instruments Plant
Much of the manufacturing of Texas Instruments equipment is done at this modern Expressway facility in Dallas, Texas

Mentathlete



Home computers. Do they send your brain to sleep – or keep your mind on its toes?

At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.

Provided, of course, it offers a whole battery of genuine mental challenges.

The Spectrum does just that.

Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!

The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.

And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.

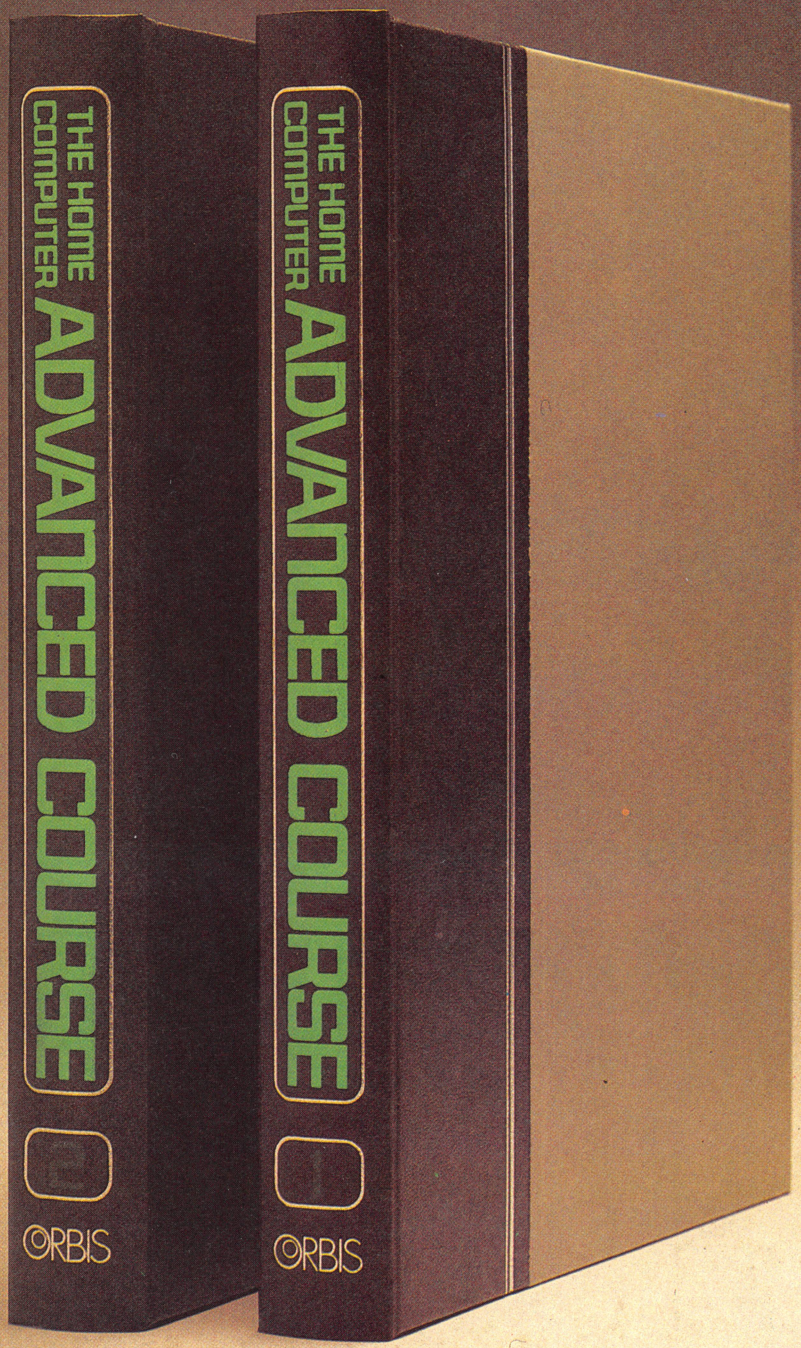
No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.

For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.

sinclair

THE HOME COMPUTER ADVANCED COURSE

WE HAVE DESIGNED BINDERS SPECIALLY TO KEEP YOUR COPIES OF THE 'ADVANCED COURSE' IN GOOD ORDER.



All you have to do is complete the reply-paid order form opposite – tick the box and post the card today – **no stamp necessary!**

By choosing a standing order, you will be sent the first volume free along with the second binder for £3.95. The invoice for this amount will be with the binder. We will then send you your binders every twelve weeks – as you need them.

Important: This offer is open only whilst stocks last and only one free binder may be sent to each purchaser who places a Standing Order. Please allow 28 days for delivery.

Overseas readers: This free binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain binders now. For details please see inside the front cover. Binders may be subject to import duty and/or local tax.

The Orbis Guarantee: If you are not entirely satisfied you may return the binder(s) to us within 14 days and cancel your Standing Order. You are then under no obligation to pay and no further binders will be sent except upon request.

PLACE A STANDING ORDER TODAY.

17012AL